Recent Progress on Boolean Beaver Triple PCG and Distributed DPF KeyGen with Active Security for

ePrint 2024/429 & 2023/845 & 2024/426

April 11, 2024 · Presented by Hongrui Cui

Quantum LWE

- 2024/555

Quantum Algorithms for Lattice Problems

Yilei Chen*

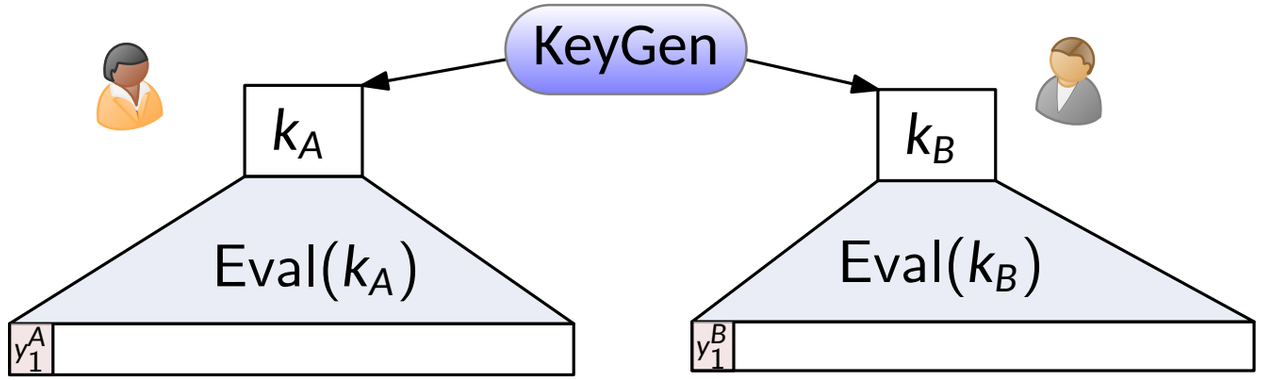
April 10, 2024

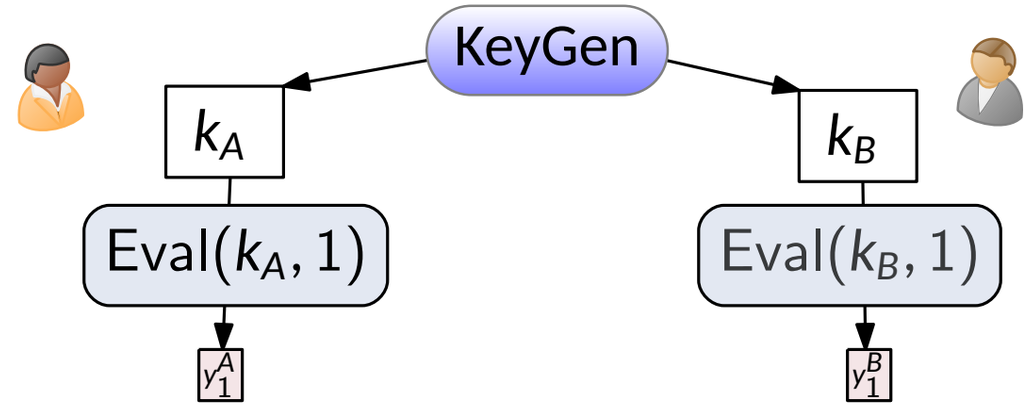
Abstract

We show a polynomial time quantum algorithm for solving the learning with errors problem (LWE) with certain polynomial modulus-noise ratios. Combining with the reductions from lattice problems to LWE shown by Regev [J.ACM 2009], we obtain polynomial time quantum algorithms for solving the decisional shortest vector problem (GapSVP) and the shortest independent vector problem (SIVP) for all n-dimensional lattices within approximation factors of $\tilde{\Omega}(n^{4.5})$. Previously, no polynomial or even subexponential time quantum algorithms were known for solving GapSVP or SIVP for all lattices within any polynomial approximation factors.

To develop a quantum algorithm for solving LWE, we mainly introduce two new techniques. First, we introduce Gaussian functions with *complex* variances in the design of quantum algorithms. In particular, we exploit the feature of the *Karst wave* in the discrete Fourier transform of complex Gaussian functions. Second, we use *windowed* quantum Fourier transform with complex Gaussian windows, which allows us to combine the information from both time and frequency domains. Using those techniques, we first convert the LWE instance into quantum states with purely imaginary Gaussian amplitudes, then convert purely imaginary Gaussian states into classical linear equations over the LWE secret and error terms, and finally solve the linear system of equations using Gaussian elimination. This gives a polynomial time quantum algorithm for solving LWE.

Introduction





Correlation Examples

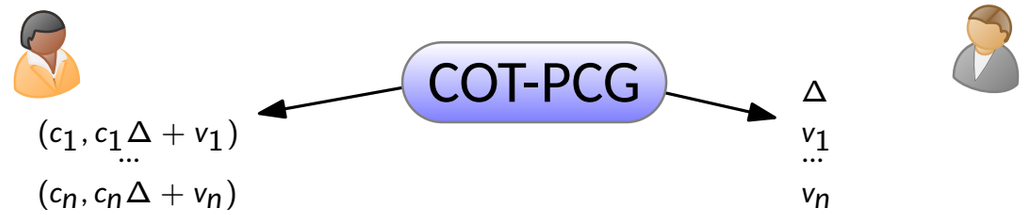
- $y_1^A = y_1^B$
- $y_1^A = (w_1, \Delta), y_1^B = (u_1, v_1),$ s.t. $w_1 = v_1 + u_1 \cdot \Delta$
- $y_1^A + y_1^B = (a, b, a \cdot b)$
- $y_1^A + y_1^B = (a, b, a \cdot b, a\Delta, b\Delta, ab\Delta)$

Motivation of This Line of Work

- **Silent** generation/PCG of Beaver triples over $\mathbb{F}_2$
- Application 1: Silent GMW Preprocessing
- Application 2: GC-PCG

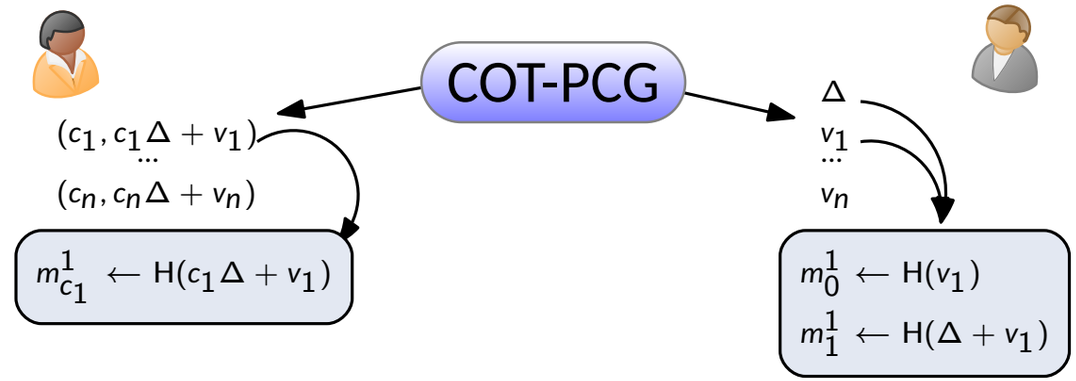
Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT



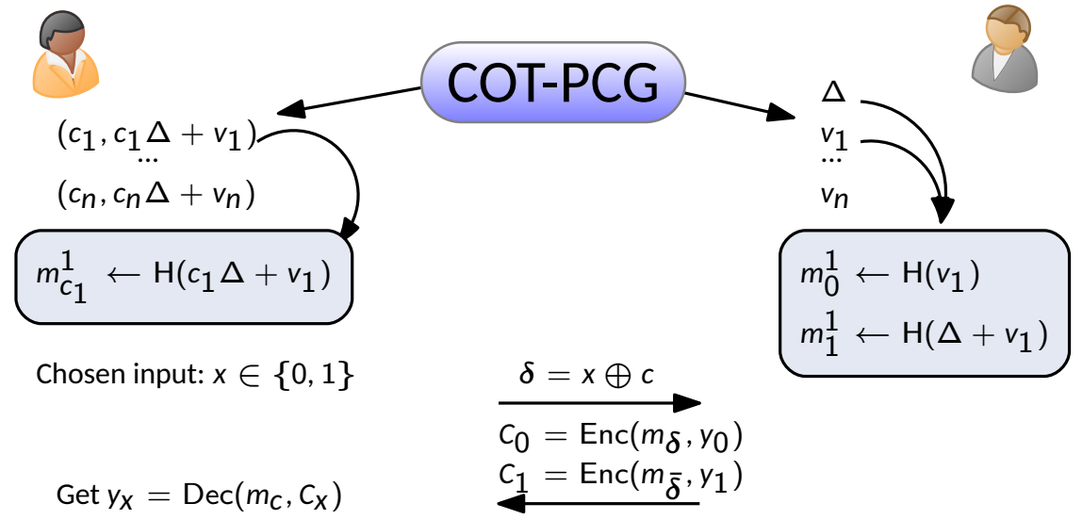
Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT



Backgrounds on MPC/2PC

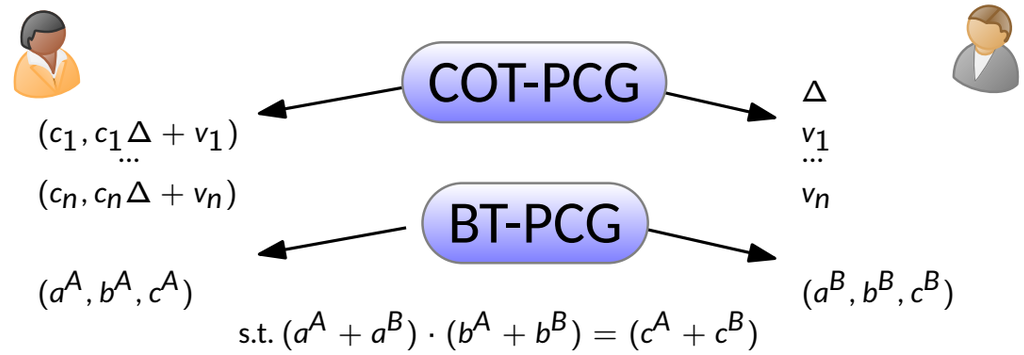
- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT



Backgrounds on MPC/2PC

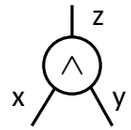
- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT

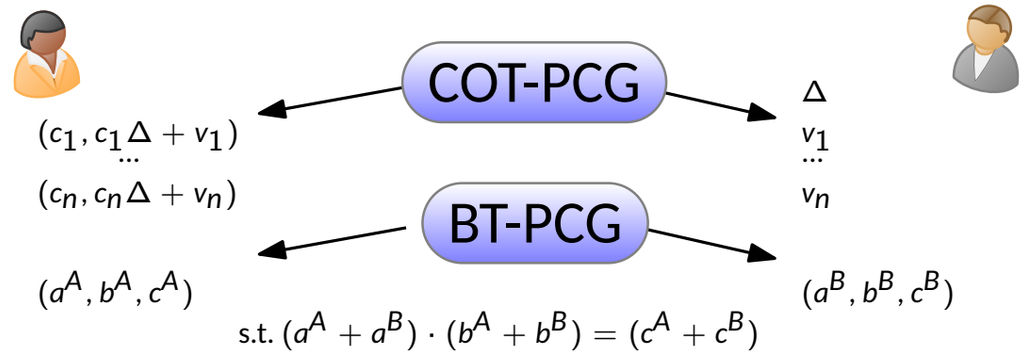
- Beaver Triple



Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple



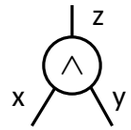


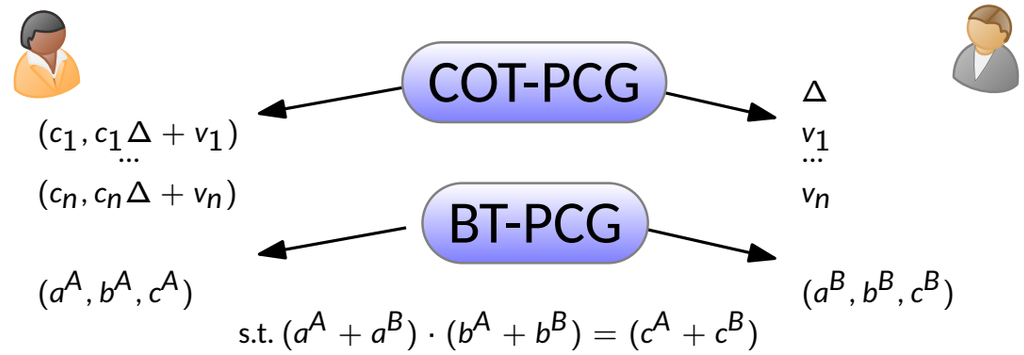
- Alice has $x \in \{0,1\}$ and samples x^A, x^B s.t. $x^A \oplus x^B = x$, sends x^B to Bob
- Bob has $y \in \{0,1\}$ and samples y^A, y^B s.t. $y^A \oplus y^B = y$, sends y^B to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple





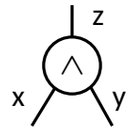
- Alice has $x \in \{0,1\}$ and samples x^A, x^B s.t. $x^A \oplus x^B = x$, sends x^B to Bob
- Bob has $y \in \{0,1\}$ and samples y^A, y^B s.t. $y^A \oplus y^B = y$, sends y^B to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

$$x \cdot y = (\underbrace{x \oplus a}_{e} \oplus a) \cdot (\underbrace{y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple



- Alice has $x \in \{0,1\}$ and samples x^A, x^B s.t. $x^A \oplus x^B = x$, sends x^B to Bob
- Bob has $y \in \{0,1\}$ and samples y^A, y^B s.t. $y^A \oplus y^B = y$, sends y^B to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

- Active security with SPDZ-style authentication

COT-PCG

$(c_1, c_1\Delta + v_1)$ … $(c_n, c_n\Delta + v_n)$ — Δ; v_1 … v_n

BT-PCG

(a^A, b^A, c^A) — (a^B, b^B, c^B)

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

$$x \cdot y = \underbrace{(x \oplus a}_{e} \oplus a) \cdot \underbrace{(y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

ABT-PCG

(a^A, b^A, c^A) — (a^B, b^B, c^B)

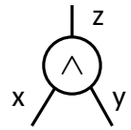
$(\Delta^A, \mathrm{M}[a^A], \mathrm{M}[b^A], \mathrm{M}[c^A])$ — $(\Delta^B, \mathrm{M}[a^B], \mathrm{M}[b^B], \mathrm{M}[c^B])$

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

$(u^A + u^B) \cdot (\Delta^A + \Delta^B) = \mathrm{M}[u^A] + \mathrm{M}[u^B]$ over $\mathbb{F}_{2^\lambda}$

Backgrounds on MPC/2PC

- Oblivious Transfer from Correlated OT
- **C**orrelation **R**esistant hash to get random OT
- Receiver sends diff. to get OT
- Beaver Triple



- Alice has $x \in \{0,1\}$ and samples x^A, x^B s.t. $x^A \oplus x^B = x$, sends x^B to Bob
- Bob has $y \in \{0,1\}$ and samples y^A, y^B s.t. $y^A \oplus y^B = y$, sends y^B to Alice
- Parties **open** $e = x \oplus a, f = y \oplus b$

COT-PCG

$(c_1, c_1\Delta + v_1)$
$\cdots$
$(c_n, c_n\Delta + v_n)$

Δ
v_1
$\cdots$
v_n

BT-PCG

(a^A, b^A, c^A) $\quad$ (a^B, b^B, c^B)

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

$$x \cdot y = \underbrace{(x \oplus a}_{e} \oplus a) \cdot \underbrace{(y \oplus b}_{f} \oplus b) = ef \oplus eb \oplus af \oplus \underbrace{ab}_{c}$$

$$[\![z]\!] = ef \oplus e[\![b]\!] \oplus f[\![a]\!] \oplus [\![c]\!]$$

ABT-PCG

(a^A, b^A, c^A) $\quad$ (a^B, b^B, c^B)

$(\Delta^A, \mathrm{M}[a^A], \mathrm{M}[b^A], \mathrm{M}[c^A])$ $\quad$ $(\Delta^B, \mathrm{M}[a^B], \mathrm{M}[b^B], \mathrm{M}[c^B])$

s.t. $(a^A + a^B) \cdot (b^A + b^B) = (c^A + c^B)$

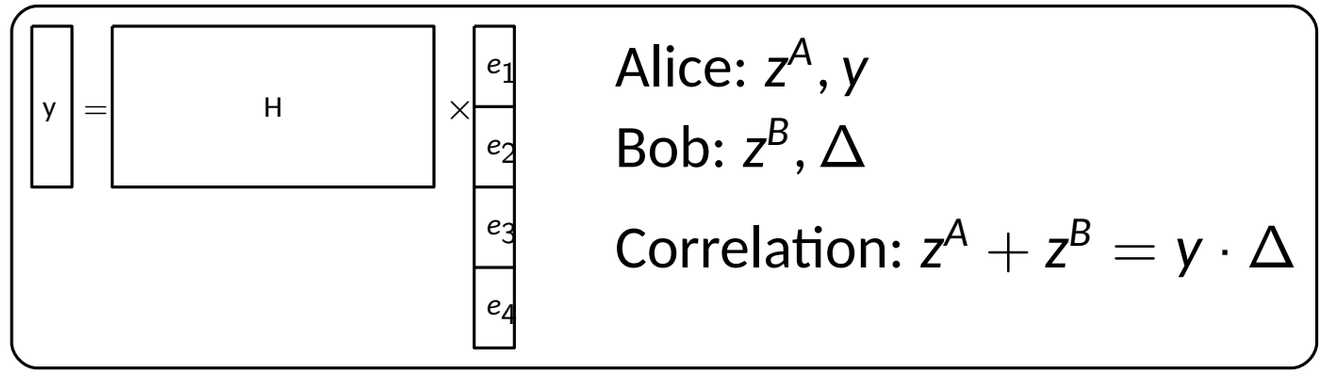
$(u^A + u^B) \cdot (\Delta^A + \Delta^B) = \mathrm{M}[u^A] + \mathrm{M}[u^B]$ over $\mathbb{F}_{2^\lambda}$

- Active security with SPDZ-style authentication
- IT-MAC has additive-homomorphism
- **open** = reveal + check
- $\mathrm{reveal}(x) = \text{send } x^A, x^B$
- $\mathrm{check}(x) = \text{check } \mathrm{M}[x^A] + \mathrm{M}[x^B] = (x^A + x^B)(\Delta^A + \Delta^B)$

Paradigm for PCG

Paradigm for COT/sVOLE PCG

- Generate **sparse** correlations
- Compress with linear map (LPN)



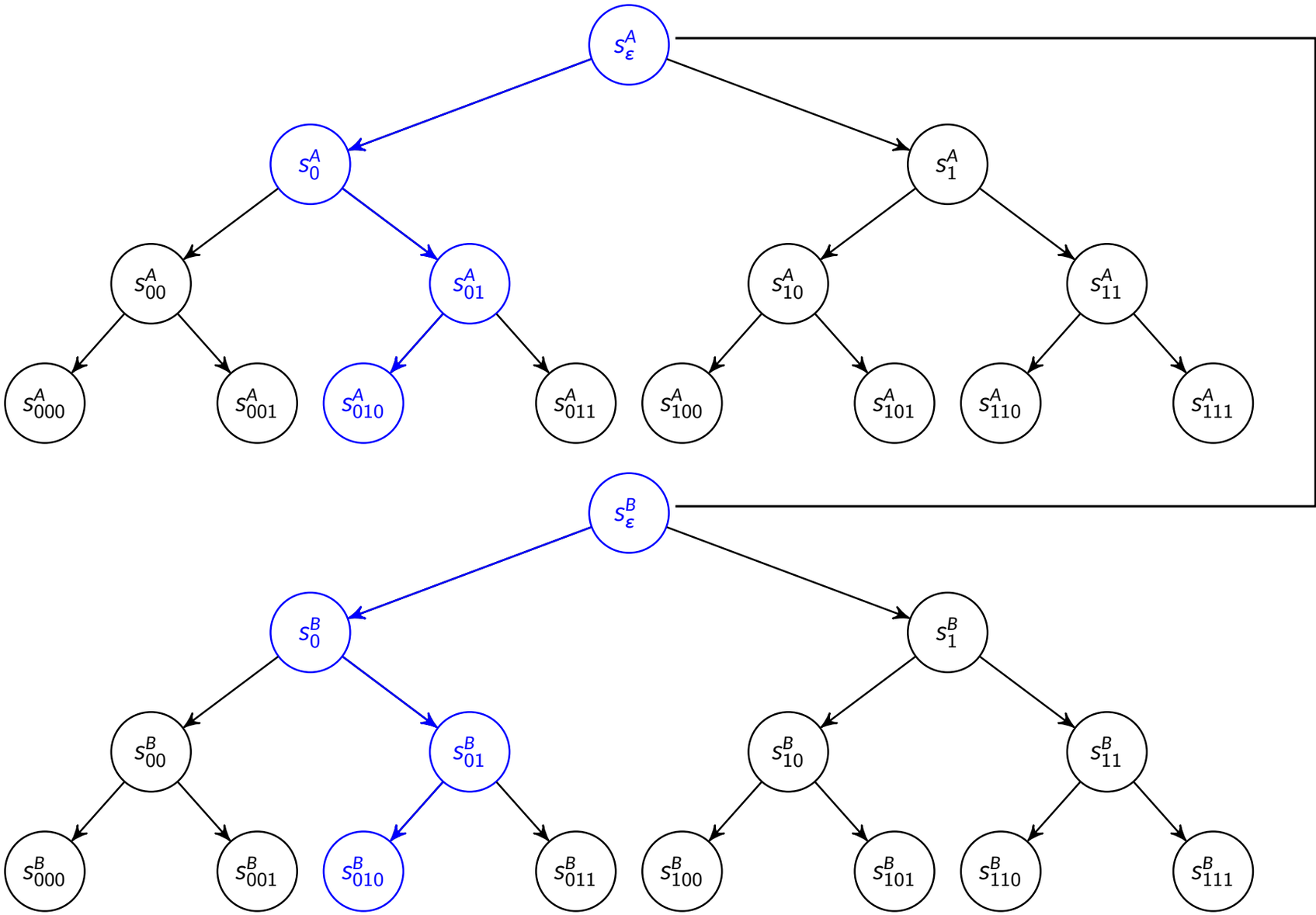
FSS for DPF/RDPF

- Input: $[\alpha], [\beta]$
- Output: (k^A, k^B)
- Correlation: $\text{Eval}(k^A, x) + \text{Eval}(k^B, x) = \begin{cases} \beta & x = \alpha \\ 0 & o.w. \end{cases}$

SPFSS: **S**um of single **P**oint **FSS** (also called Multi-Point FSS)

- For a t-sparse noise, generate t-pairs of DPF FSS keys
- Full domain evaluation gives us $\mathbf{e}$ or $\mathbf{e} \cdot \Delta$
- $\text{FullEval}(k^A) + \text{FullEval}(k^B) = \mathbf{e} \cdot \Delta \rightarrow$ Left multiply by H gives us the desired correlation.

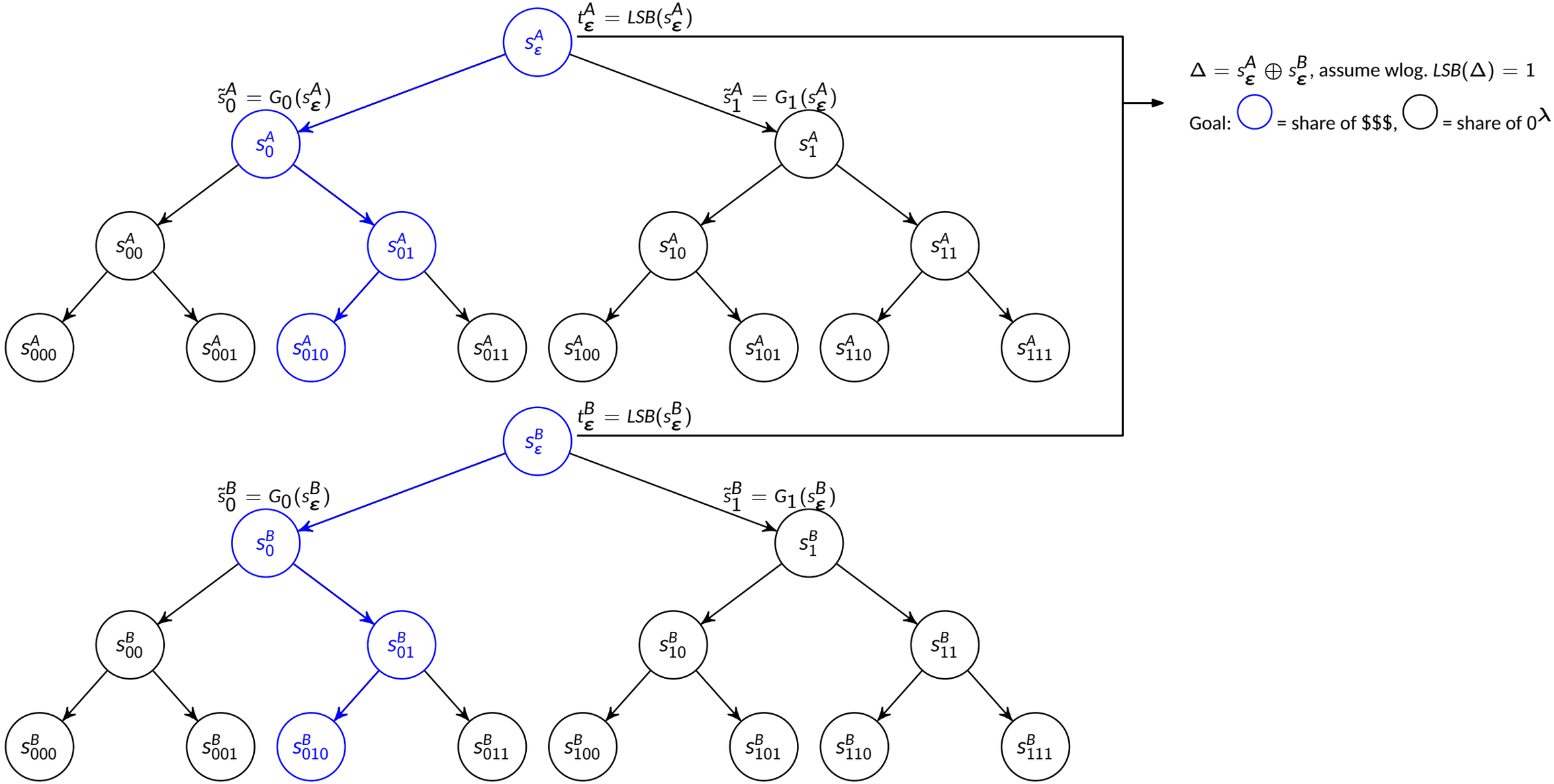
More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



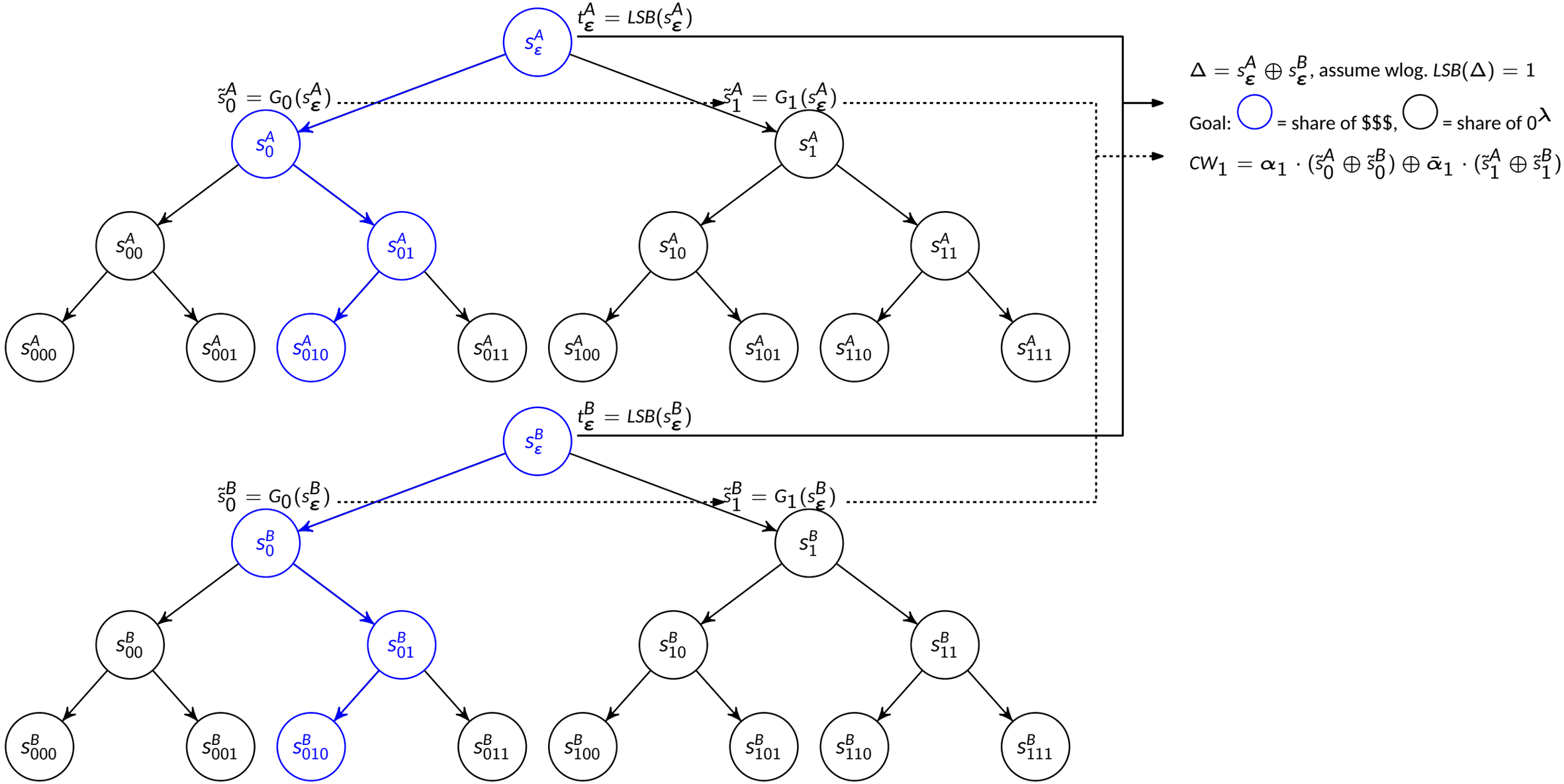
$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of \$\$\$, ◯ = share of 0^λ

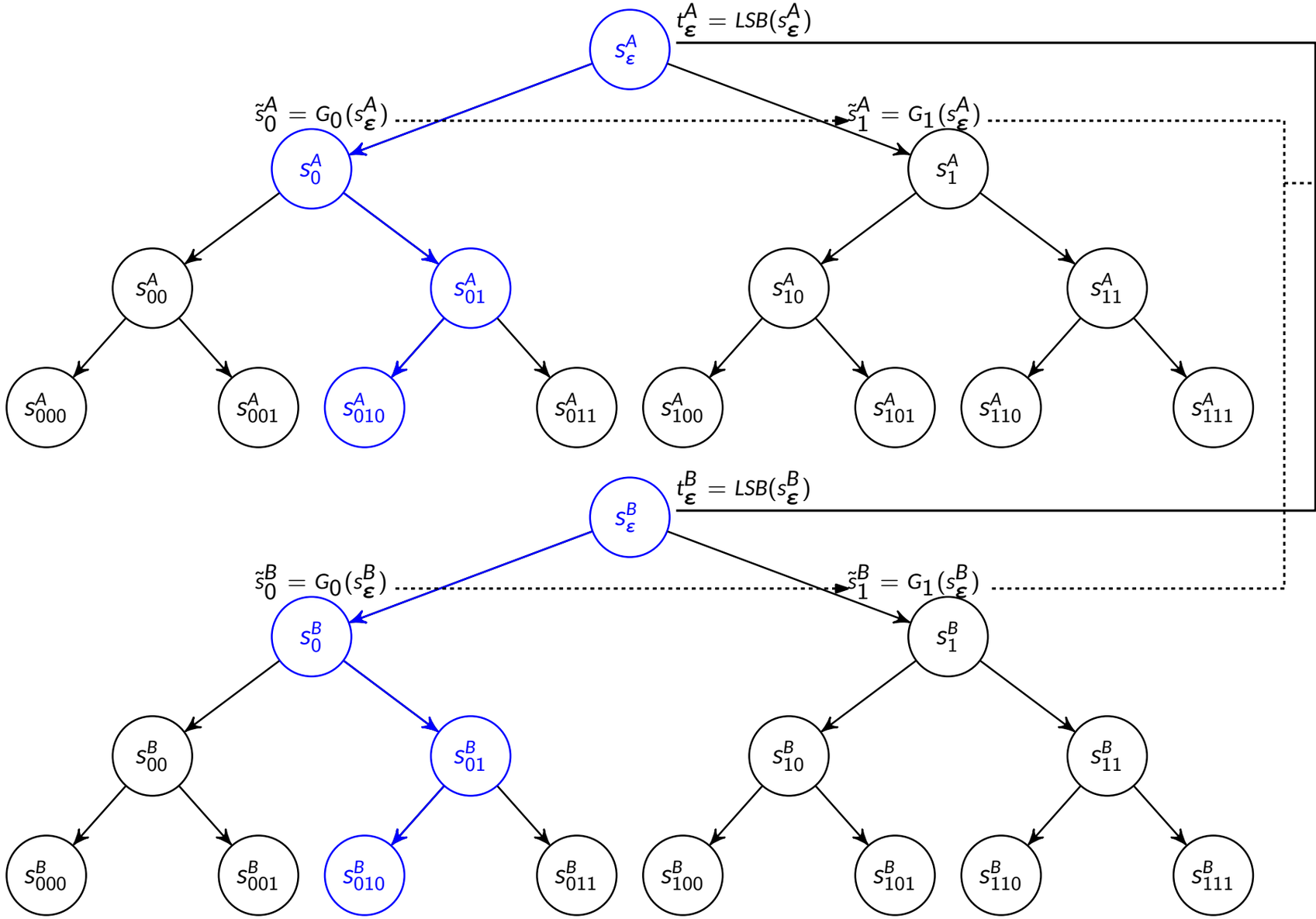
More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

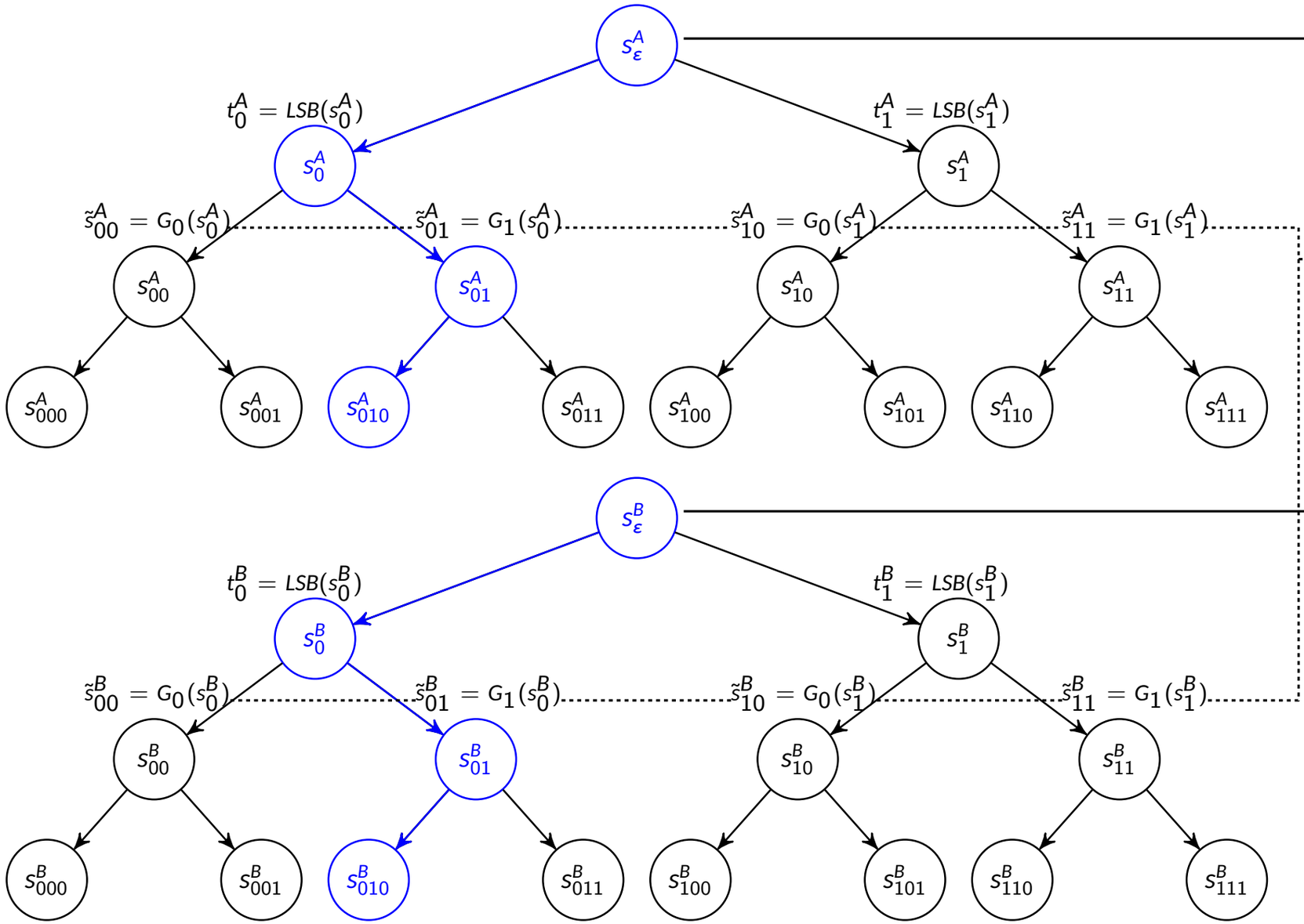
Goal: ◯ = share of $\$\$$, ◯ = share of 0^λ

$CW_1 = \alpha_1 \cdot (\tilde{s}^A_0 \oplus \tilde{s}^B_0) \oplus \bar{\alpha}_1 \cdot (\tilde{s}^A_1 \oplus \tilde{s}^B_1)$

$s^A_0 = \tilde{s}^A_0 \oplus t^A_\varepsilon \cdot CW_1, s^A_1 = \tilde{s}^A_1 \oplus t^A_\varepsilon \cdot CW_1$

$s^B_0 = \tilde{s}^B_0 \oplus t^B_\varepsilon \cdot CW_1, s^B_1 = \tilde{s}^B_1 \oplus t^B_\varepsilon \cdot CW_1$

More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



$\Delta = s^A_{\boldsymbol{\varepsilon}} \oplus s^B_{\boldsymbol{\varepsilon}}$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of \$\$, ◯ = share of 0^λ

$$CW_2 = \alpha_2 \cdot (\tilde{s}^A_{00} \oplus \tilde{s}^B_{00} \oplus \tilde{s}^A_{10} \oplus \tilde{s}^B_{10}) \oplus \bar{\alpha}_2 \cdot (\tilde{s}^A_{01} \oplus \tilde{s}^B_{01} \oplus \tilde{s}^A_{11} \oplus \tilde{s}^B_{11})$$

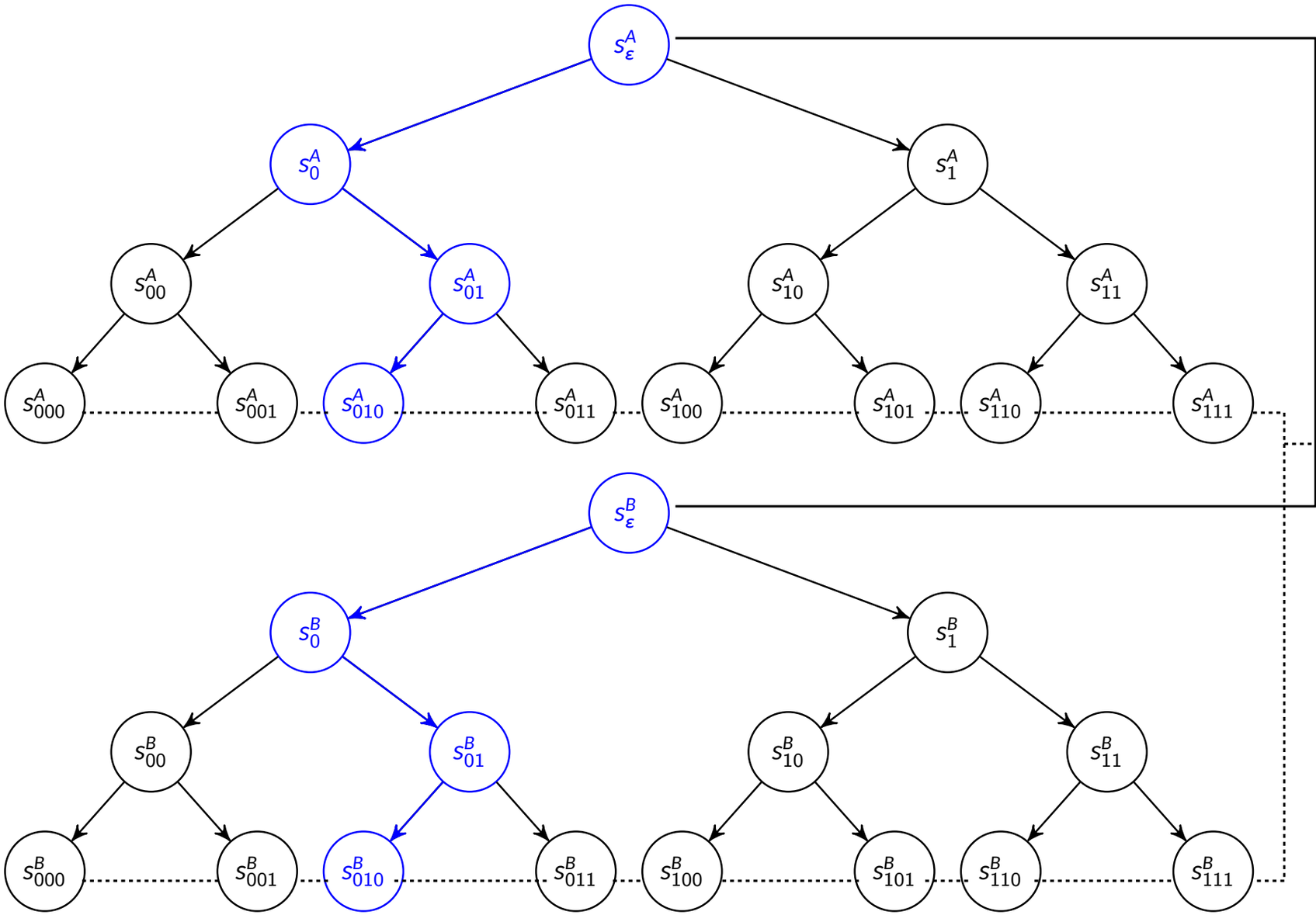
$$s^A_{00} = \tilde{s}^A_{00} \oplus t^A_0 \cdot CW_2, s^A_{01} = \tilde{s}^A_{01} \oplus t^A_0 \cdot CW_2$$

$$s^A_{10} = \tilde{s}^A_{10} \oplus t^A_1 \cdot CW_2, s^A_{11} = \tilde{s}^A_{11} \oplus t^A_1 \cdot CW_2$$

$$s^B_{00} = \tilde{s}^B_{00} \oplus t^B_0 \cdot CW_2, s^B_{01} = \tilde{s}^B_{01} \oplus t^B_0 \cdot CW_2$$

$$s^B_{10} = \tilde{s}^B_{10} \oplus t^B_1 \cdot CW_2, s^B_{11} = \tilde{s}^B_{11} \oplus t^B_1 \cdot CW_2$$

More Details on DPF FSS, $\alpha = 010, \beta = \Delta$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

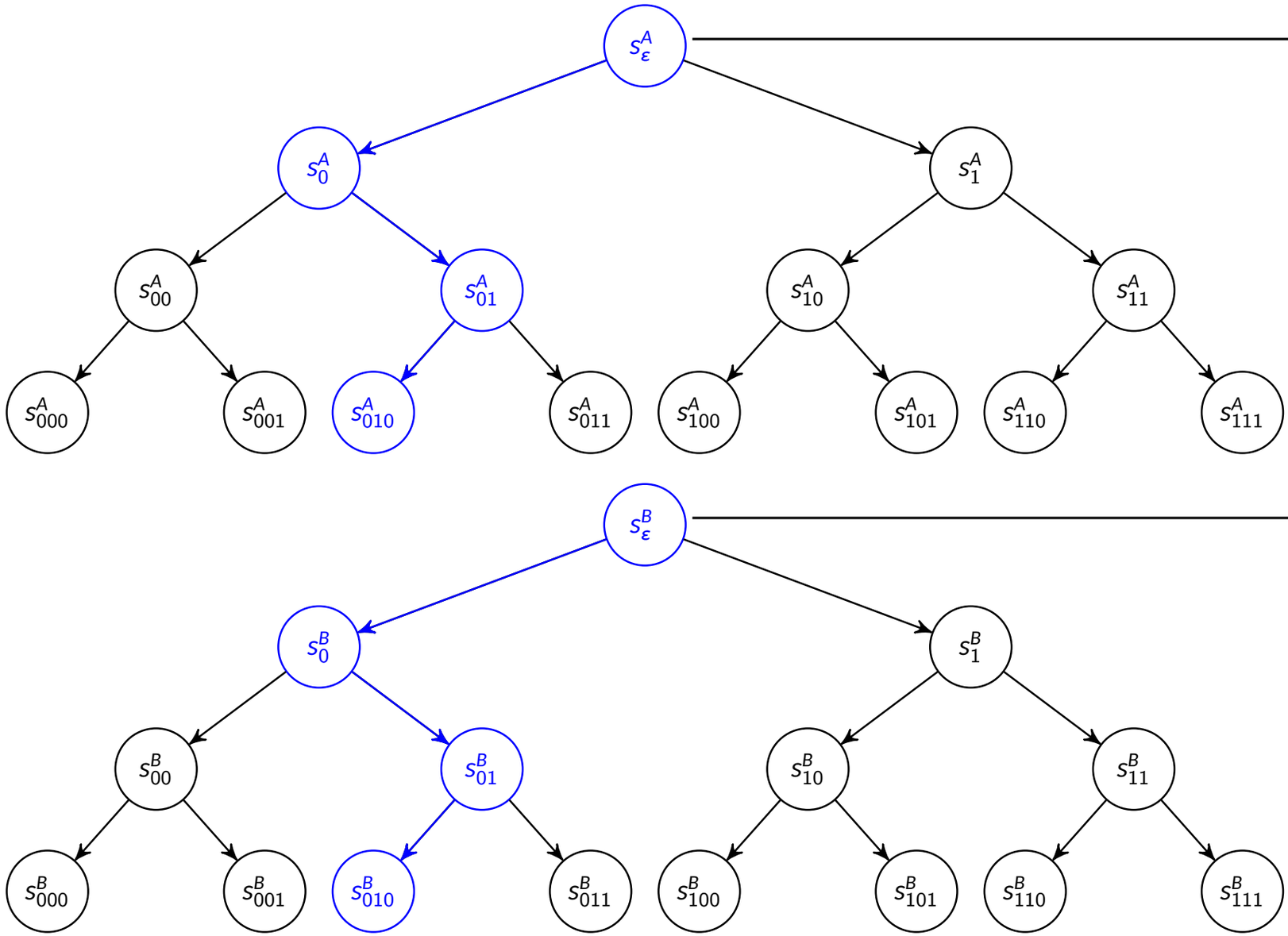
Goal: ◯ = share of \$\$\$, ◯ = share of 0^λ

CW_1, CW_2, CW_3

$CW_{\text{out}} = \sum s^A_{***} \oplus \sum s^B_{***} \oplus \Delta$

$z^A_{idx} = s^A_{idx} \oplus t^A_{idx} \cdot CW_{out}$
$z^B_{idx} = s^B_{idx} \oplus t^B_{idx} \cdot CW_{out}$ for $idx \in \{0,1\}^3$

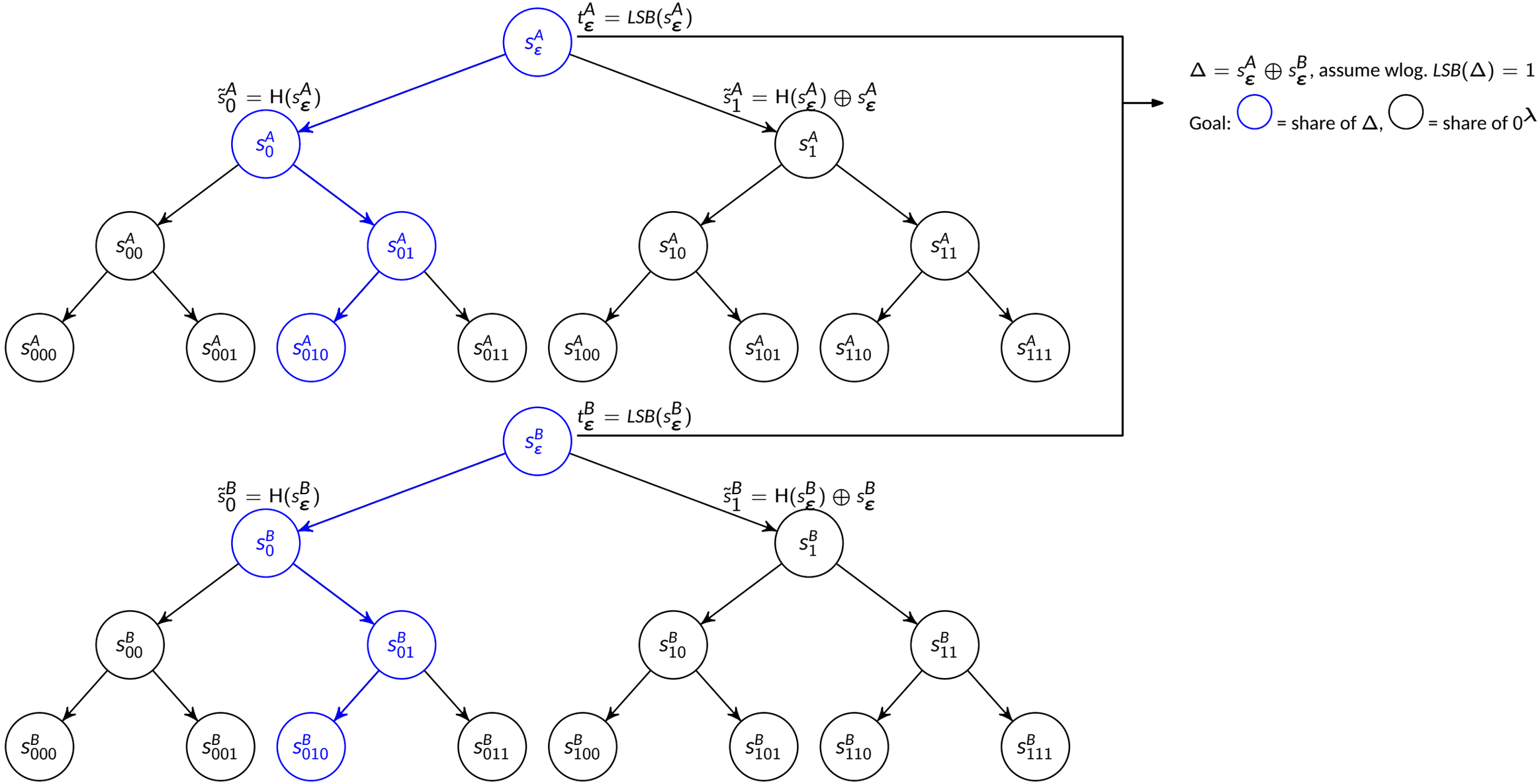
Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



$\Delta = s^A_{\boldsymbol{\varepsilon}} \oplus s^B_{\boldsymbol{\varepsilon}}$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of Δ, ◯ = share of 0^λ

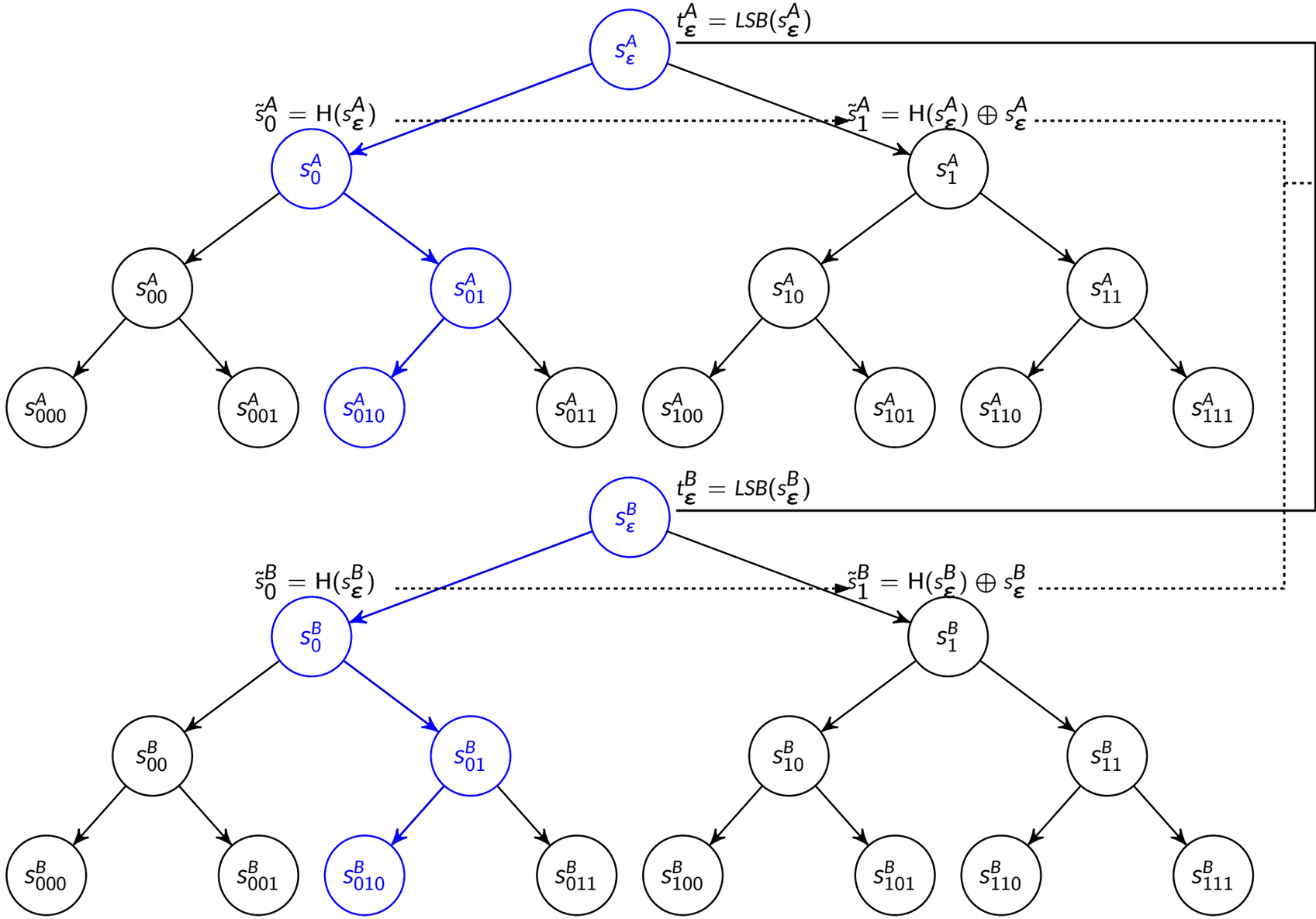
Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of Δ, ◯ = share of 0^λ

Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

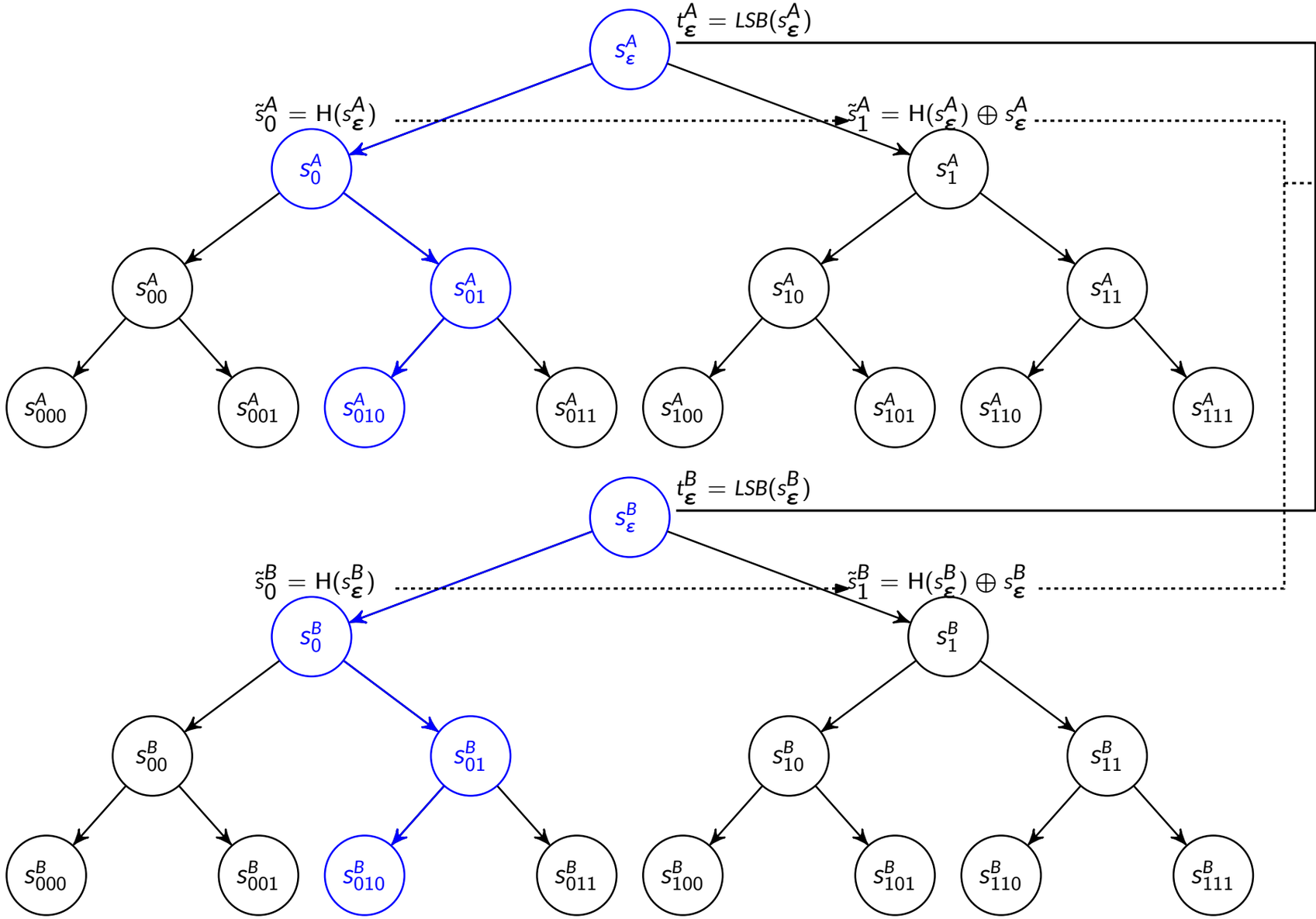
Goal: ◯ = share of Δ, ◯ = share of 0^λ

$\tilde{s}^A_0 \oplus \tilde{s}^A_1 = \tilde{s}^A_\varepsilon$ $\qquad \tilde{s}^A_\varepsilon \oplus \tilde{s}^B_\varepsilon = \Delta$

$\tilde{s}^B_0 \oplus \tilde{s}^B_1 = \tilde{s}^B_\varepsilon$

$cw_1 = \alpha_1 \cdot (\tilde{s}^A_0 \oplus \tilde{s}^B_0) \oplus \bar{\alpha}_1 \cdot (\tilde{s}^A_1 \oplus \tilde{s}^B_1)$

Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of Δ, ◯ = share of 0^λ

$\tilde{s}^A_0 \oplus \tilde{s}^A_1 = \tilde{s}^A_\varepsilon$ $\qquad \tilde{s}^A_\varepsilon \oplus \tilde{s}^B_\varepsilon = \Delta$

$\tilde{s}^B_0 \oplus \tilde{s}^B_1 = \tilde{s}^B_\varepsilon$

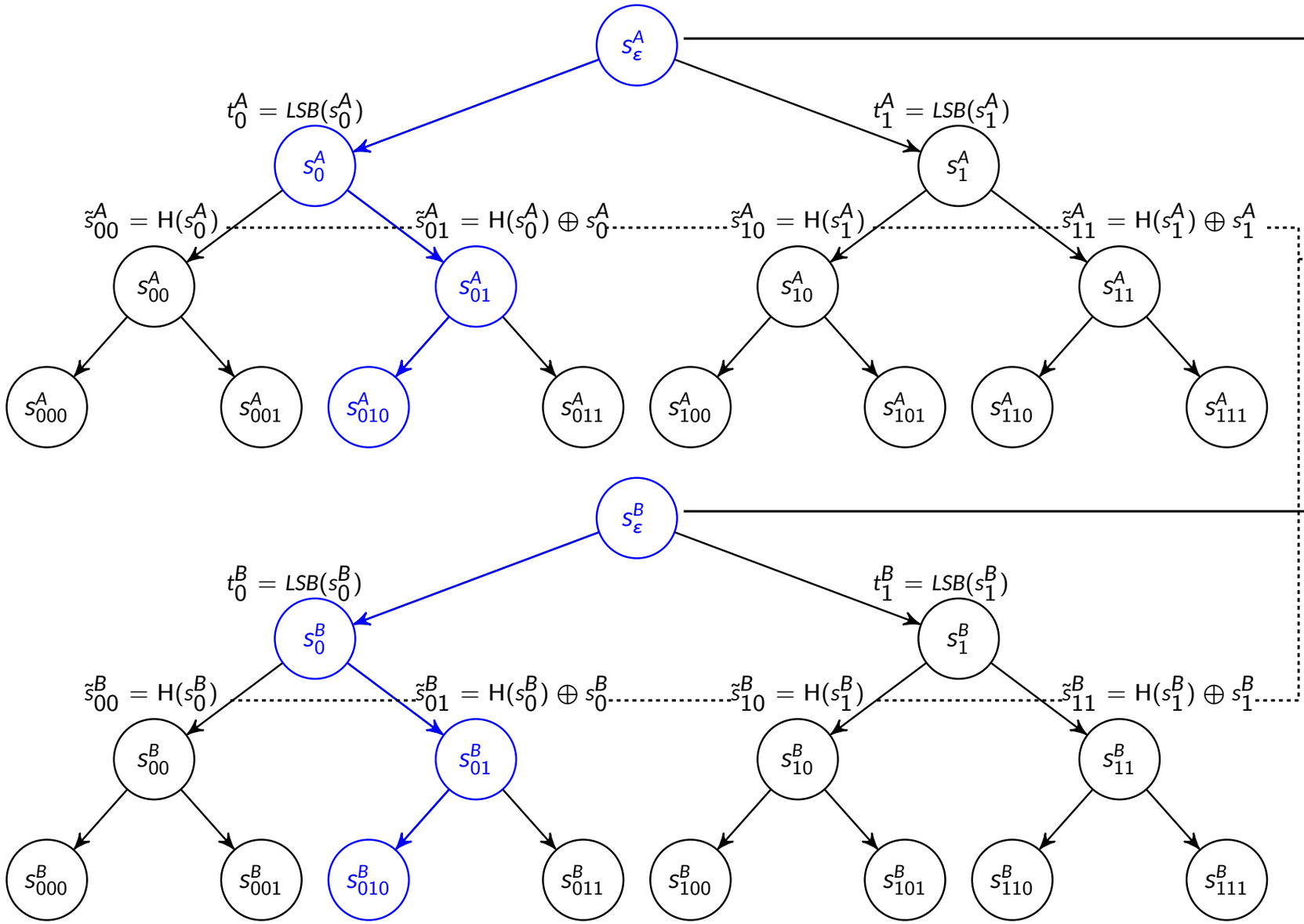
$$CW_1 = \alpha_1 \cdot (\tilde{s}^A_0 \oplus \tilde{s}^B_0) \oplus \bar{\alpha}_1 \cdot (\tilde{s}^A_1 \oplus \tilde{s}^B_1)$$
$$= (\tilde{s}^A_1 \oplus \tilde{s}^B_1) \oplus \alpha_1 \cdot \Delta$$

$s^A_0 = \tilde{s}^A_0 \oplus t^A_\varepsilon \cdot CW_1, s^A_1 = \tilde{s}^A_1 \oplus t^A_\varepsilon \cdot CW_1$

$s^B_0 = \tilde{s}^B_0 \oplus t^B_\varepsilon \cdot CW_1, s^B_1 = \tilde{s}^B_1 \oplus t^B_\varepsilon \cdot CW_1$

$s^A_0 \oplus s^B_0 = \Delta$

Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

Goal: ◯ = share of Δ, ◯ = share of 0^λ

$$CW_2 = \alpha_2 \cdot (\tilde{s}^A_{00} \oplus \tilde{s}^B_{00} \oplus \tilde{s}^A_{10} \oplus \tilde{s}^B_{10}) \oplus \bar{\alpha}_2 \cdot (\tilde{s}^A_{01} \oplus \tilde{s}^B_{01} \oplus \tilde{s}^A_{11} \oplus \tilde{s}^B_{11})$$

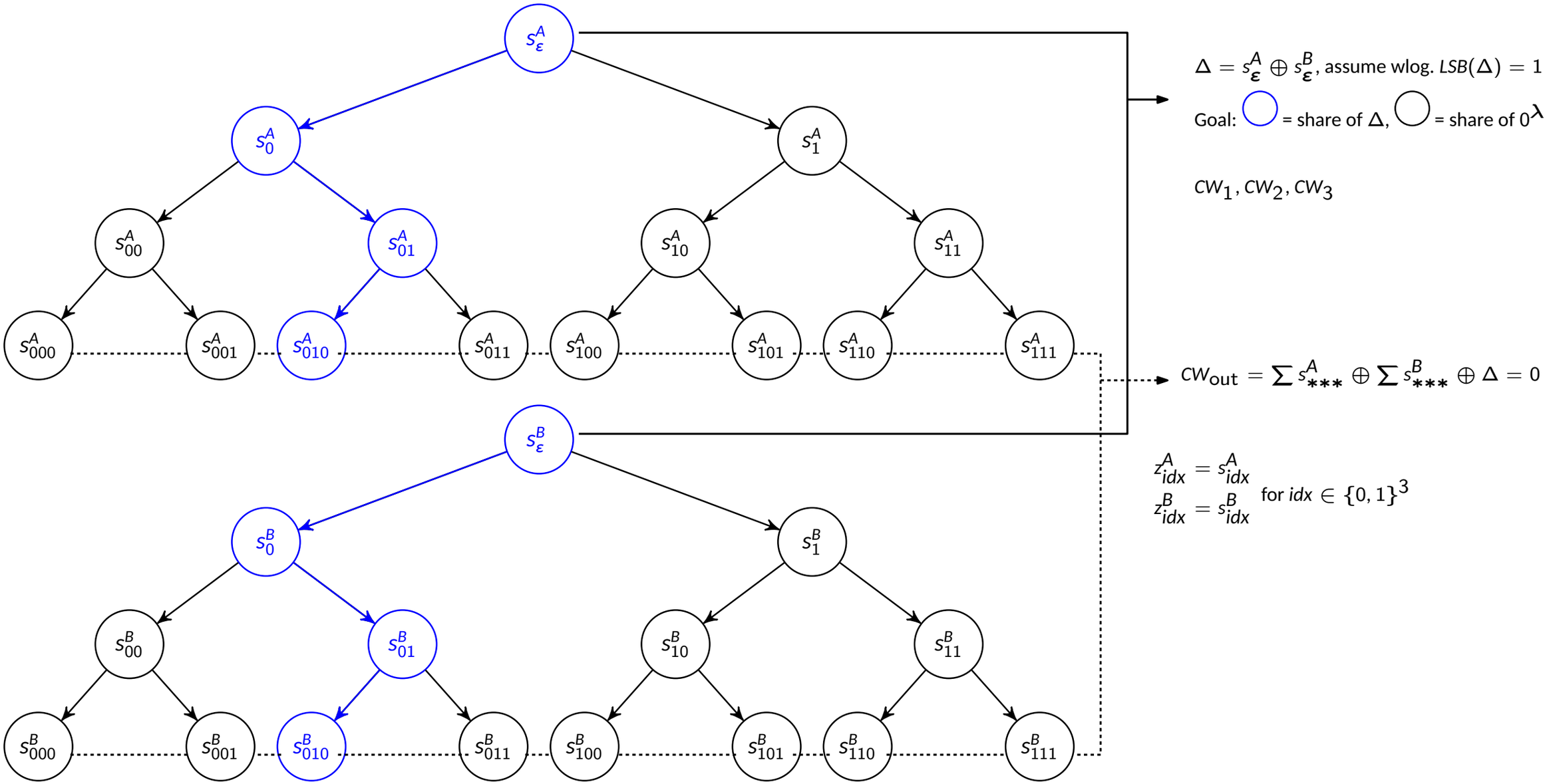
$s^A_{00} = \tilde{s}^A_{00} \oplus t^A_0 \cdot CW_2, s^A_{01} = \tilde{s}^A_{01} \oplus t^A_0 \cdot CW_2$

$s^A_{10} = \tilde{s}^A_{10} \oplus t^A_1 \cdot CW_2, s^A_{11} = \tilde{s}^A_{11} \oplus t^A_1 \cdot CW_2$

$s^B_{00} = \tilde{s}^B_{00} \oplus t^B_0 \cdot CW_2, s^B_{01} = \tilde{s}^B_{01} \oplus t^B_0 \cdot CW_2$

$s^B_{10} = \tilde{s}^B_{10} \oplus t^B_1 \cdot CW_2, s^B_{11} = \tilde{s}^B_{11} \oplus t^B_1 \cdot CW_2$

Half-Tree, $\alpha = 010, \beta = \Delta, LSB(\Delta) = 1$



Motivations

Key problem with “Quadratic” correlation

- Quadratic computation blow-up
- Consider $10^6 \rightarrow 10^{12}$

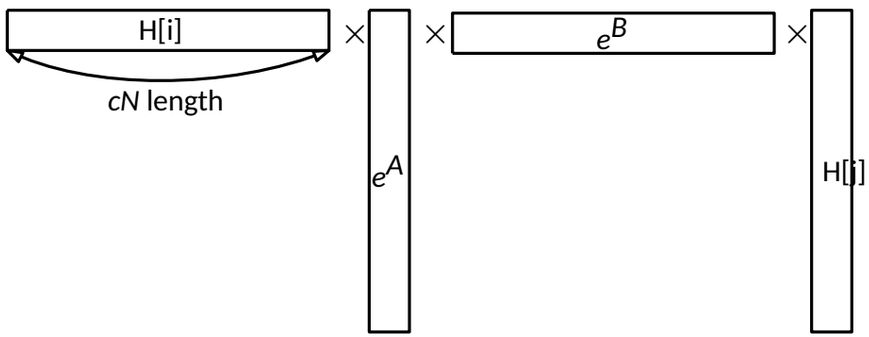
Alice: z^A, y^A, x^A

Bob: z^B, y^B, x^B

$z^A + z^B = (x^A + x^B) \cdot (y^A + y^B)$

Consider $x^A[i] \cdot y^B[j] = \langle H[i], e^A \rangle \cdot \langle H[j], e^B \rangle$

Let $H \in \mathbb{F}_p^{N \times cN}$, $|e| = t$.



For regular LPN over $\mathbb{F}_p$, $H \leftarrow \mathbb{F}_p^{N \times cN}$, expected $O(c^2N^2)$ work

Previous Solutions

BCGIKS20

- **Ring-LPN**: Replace $\langle H, e \rangle$ with $\langle a(X), e(X) \rangle$ for $a(X), e(X) \in (\mathbb{F}_q[X]/(f(X)))^c$
- Now evaluating cross-term requires $O(c^2 N \log N) = \tilde{O}(N)$ work (with FFT)
- The resulting polynomial $\langle a \otimes a, e^A \otimes e^B \rangle$ is isomorphic to $\mathbb{F}_q^N$
- **CRT** requires $q > N$

BCGIKS20 (FOCS'20)

- **VD-LPN**

BCGIKRS22

- **EA-LPN** Replace $\langle H, e \rangle$ with $\langle E \cdot A, e \rangle$ for c-sparse E, upper-triangular A
- Now evaluating cross-term requires $O(c^2 t^2 N)$ work
- Requires further cryptanalysis

BCCD23

- **QA-SD** Replace univariate polynomial in Ring-LPN with multivariate polynomial
- Generate Beaver triples over $\mathbb{F}_q$ for $q \geq 3$

BBCCDS24

- **QA-SD** over $\mathbb{F}_4$ implies Beaver triples over $\mathbb{F}_2$.
- FFT optimizations and implementation

Distributed Setup of PCG

Ds17

- Distributed setup of DPF keys with black-box 2PC

ZGYZYW24

- Half-tree DPF KeyGen from BDOZ-authenticated inputs and SPDZ-authenticated-payload

Ultimate Goal

- End-to-end 2PC with malicious security
- 1. Correct LPN variant
- 2. Matching $\Pi_{\text{FSS.KeyGen}}$ with malicious security

Problem with BCGIKS20 (Ring-LPN)

BCGIKS20

- **Ring-LPN**: Replace $\langle H, e \rangle$ with $\langle a(X), e(X) \rangle$ for $a(X), e(X) \in (\mathbb{F}_q[X]/(f(X)))^c$
- Now evaluating cross-term requires $O(c^2 N \log N) = \tilde{O}(N)$ work (with FFT)
- The resulting polynomial $\langle a \otimes a, e^A \otimes e^B \rangle$ is isomorphic to $\mathbb{F}_q^N$
- **CRT** requires $q > N$

Our Goal: Beaver Triple over $\mathbb{F}_2$

- Ring-LPN solution requires setting $q = 2^\rho$, incurring a ρ-time blow-up

- Beaver triple in GMW: Suppose we have $[\![x]\!], [\![y]\!]$ and we want to compute $[\![x \cdot y]\!]$
- Beaver triple: $([\![a]\!], [\![b]\!], [\![a \cdot b]\!])$
- $[\![x \cdot y]\!] = [\![(x \oplus a \oplus a) \cdot (y \oplus b \oplus b)]\!] = [\![(x \oplus a)(y \oplus b)]\!] \oplus [\![(x \oplus a)b]\!] \oplus [\![a(y \oplus b)]\!] \oplus [\![ab]\!]$

Quasi-Abelian Syndrome Decoding

$$\mathbb{F}_q[G] \stackrel{\text{def}}{=} \left\{ \sum_{g \in G} a_g g \mid a_g \in \mathbb{F}_q \right\}$$

- $G = \{1_G\}$: $\mathbb{F}_q[G] = \mathbb{F}_q$
- $G = \mathbb{Z}/n\mathbb{Z}$: $\mathbb{F}_q[G] = \mathbb{F}_q[X]/(X^n - 1)$

13.1: Finite Abelian Groups

In our investigation of cyclic groups we found that every group of prime order was isomorphic to $\mathbb{Z}_p$, where p was a prime number. We also determined that $\mathbb{Z}_{mn} \cong \mathbb{Z}_m \times \mathbb{Z}_n$ when $\gcd(m, n) = 1$. In fact, much more is true. Every finite abelian group is isomorphic to a direct product of cyclic groups of prime power order; that is, every finite abelian group is isomorphic to a group of the type

$$\mathbb{Z}_{p_1^{\alpha_1}} \times \cdots \times \mathbb{Z}_{p_n^{\alpha_n}},$$

where each p_k is prime (not necessarily distinct).

Multiplication by convolution

$$\left(\sum_{g \in G} a_g g \right) \left(\sum_{g \in G} b_g g \right) \stackrel{\text{def}}{=} \sum_{g \in G} \left(\sum_{h \in G} a_h b_{h^{-1}g} \right) g$$

(Search) QA-SD problem. Given $\mathbf{H} = (\mathbf{1} \mid \mathrm{a})$ a parity check matrix of a random systematic quasi-abelian code, a target weight $t \in \mathbb{N}$ and a syndrome $\mathbf{s} \in \mathbb{F}_q[G]$, the goal is to recover an error $\mathbf{e} = (\mathbf{e}_1 \mid \mathbf{e}_2)$ with $\mathbf{e}_i \leftarrow \mathcal{D}_t(\mathbb{F}_q[G])$ such that $\mathbf{H}\mathbf{e}^T = \mathbf{s}$, i.e. $\mathbf{e}_1 + \mathbf{a} \cdot \mathbf{e}_2 = \mathbf{s}$.

Quasi-Abelian Syndrome Decoding in PCG

Recall our goal: $z^A + z^B = (x^A + x^B) \cdot (y^A + y^B)$

- Let $\mathbf{x}^A = \langle \mathbf{a}, \mathbf{e}_0 \rangle$, $\mathbf{y}^B = \langle \mathbf{a}, \mathbf{e}_1 \rangle$ c-length **vector** inner product over $\mathbb{F}_q[G]$
- Let $\mathbf{x}^A\mathbf{y}^B = \langle \mathbf{a} \otimes \mathbf{a}, \mathbf{e}_0 \otimes \mathbf{e}_1 \rangle$
- FullEval$(\mathbf{x}^A\mathbf{y}^B)[i] = x^A[i] \cdot y^B[i]$ over $\mathbb{F}_q$

Multiplication by convolution

$$\left(\sum_{i\in[t]} a_{g_i} g_i\right)\left(\sum_{j\in[t]} b_{h_j} h_j\right) = \sum_{i,j\in[t]} a_{g_i} b_{h_j} (g_i \circ h_j)$$

Quasi-Abelian Syndrome Decoding in PCG

Recall our goal: $z^A + z^B = (x^A + x^B) \cdot (y^A + y^B)$

- Let $\mathbf{x}^A = \langle \mathbf{a}, \mathbf{e}_0 \rangle$, $\mathbf{y}^B = \langle \mathbf{a}, \mathbf{e}_1 \rangle$ c-length **vector** inner product over $\mathbb{F}_q[G]$
- Let $\mathbf{x}^A\mathbf{y}^B = \langle \mathbf{a} \otimes \mathbf{a}, \mathbf{e}_0 \otimes \mathbf{e}_1 \rangle$
- FullEval$(\mathbf{x}^A\mathbf{y}^B)[i] = x^A[i] \cdot y^B[i]$ over $\mathbb{F}_q$

Multiplication by convolution

$$\left(\sum_{i\in[t]} a_{g_i} g_i\right)\left(\sum_{j\in[t]} b_{h_j} h_j\right) = \sum_{i,j\in[t]} a_{g_i} b_{h_j} (g_i \circ h_j)$$

- Use c^2t^2 DPF FSS to share $\mathbf{e}_0 \otimes \mathbf{e}_1$
- Locally evaluate the additive share of $\mathbf{e}_0 \otimes \mathbf{e}_1$ and convert them into shares over $\mathbb{F}_q[G]$
- Perform $\mathbb{F}_q[G]$ inner product
- Perform FullEval to get final output

Choice of G

- The most interesting case is $\mathbb{F}_q = \mathbb{F}_2$
- However, when $q = 2$, $G = \{1_G\} \otimes ... \otimes \{1_G\}$ has order 1
- **FOLEAGE** sets $q = 4$, $G = (\mathbb{Z}/3\mathbb{Z})^n$
- $\mathbb{F}_q[G] \cong \mathbb{F}_q[X_1, ..., X_n]/(X_1^3 - 1, ..., X_n^3 - 1) \cong \mathbb{F}_q^{3^n}$

Why $\mathbb{F}_4$:

Let $\left(\llbracket a \rrbracket^4, \llbracket b \rrbracket^4, \llbracket ab \rrbracket^4\right)$ be a *Beaver* triple over $\mathbb{F}_4$. Writing $x = x(0) + \theta \cdot x(1)$ for any $x \in \mathbb{F}_4$, with θ a root of the polynomial $X^2 + X + 1$ (hence $\theta^2 = \theta + 1$), we have

$$a \cdot b = a(0)b(0) + a(1)b(1) + \theta \cdot (a(0)b(1) + a(1)b(0) + a(1)b(1))$$

$$\rightarrow (ab)(0) = a(0)b(0) + a(1)b(1)$$

2-Party Case: Let $(a, b, \llbracket ab \rrbracket^4)$ be an *OLE* triple

$$(a \cdot b)(0) = \llbracket ab \rrbracket_A^4(0) + \llbracket ab \rrbracket_B^4(0) = a(0)b(0) + a(1)b(1),$$

$$\underbrace{a(0)a(1) + \llbracket ab \rrbracket_A^4(0)}_{\text{known by } A} + \underbrace{b(0)b(1) + \llbracket ab \rrbracket_B^4(0)}_{\text{known by } B} = \underbrace{(a(0) + b(1))}_{\text{shared by } A,B} \cdot \underbrace{(a(1) + b(0))}_{\text{shared by } A,B}.$$

Optimized Distributed KeyGen

Protocol $\Pi_{\mathsf{rDPF\text{-}CW}}$

PARAMETERS:

- Party $\sigma \in \{0,1\}$ has input $[\![\alpha_i]\!]_\sigma \in \mathbb{F}_3, r_i^\sigma \in \{0,1\}^\lambda, (s_{i,j}^\sigma \| t_{i,j}^\sigma)_{j\in\{0,1,2\}} \in \{0,1\}^{3(\lambda+1)}$.
- An instantiation of chosen $\binom{1}{3}$-OT.

PROTOCOL:

For each party $\sigma \in \{0,1\}$:

1: Sample $z^\sigma \leftarrow_R \{0,1\}^{3(\lambda+1)}$.

2: Define

$$\begin{aligned}
\mathbf{C}_0^\sigma &:= (r_i^\sigma \oplus s_{i,0}^\sigma \| (t_{i,0}^\sigma \oplus \sigma),\ s_{i,1}^\sigma \| t_{i,1}^\sigma,\ s_{i,2}^\sigma \| t_{i,2}^\sigma) \oplus z^\sigma \quad \triangleright\ [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 0\\
\mathbf{C}_1^\sigma &:= (s_{i,0}^\sigma \| t_{i,0}^\sigma,\ r_i^\sigma \oplus s_{i,1}^\sigma \| (t_{i,1}^\sigma \oplus \sigma),\ s_{i,2}^\sigma \| t_{i,2}^\sigma) \oplus z^\sigma \quad \triangleright\ [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 1\\
\mathbf{C}_2^\sigma &:= (s_{i,0}^\sigma \| t_{i,0}^\sigma,\ s_{i,1}^\sigma \| t_{i,1}^\sigma,\ r_i^\sigma \oplus s_{i,2}^\sigma \| (t_{i,2}^\sigma \oplus \sigma)) \oplus z^\sigma \quad \triangleright\ [\![\mathsf{CW}_i]\!]_\sigma \text{ when } \alpha_i = 2\\
\mathbf{M}_0^\sigma &:= (\mathbf{C}_0^\sigma, \mathbf{C}_1^\sigma, \mathbf{C}_2^\sigma),\ \mathbf{M}_1^\sigma := (\mathbf{C}_1^\sigma, \mathbf{C}_2^\sigma, \mathbf{C}_0^\sigma),\ \mathbf{M}_2^\sigma := (\mathbf{C}_2^\sigma, \mathbf{C}_0^\sigma, \mathbf{C}_1^\sigma)
\end{aligned}$$

3: Invoke $\binom{1}{3}$-OT with party $\bar{\sigma}$ as follows:

- Party $\bar{\sigma}$ plays the role of the sender with inputs $\mathbf{M}^{\bar{\sigma}}_{[\![\alpha_i]\!]_{\bar{\sigma}}}$.
- Party σ plays the role of the receiver and inputs $[\![\alpha_i]\!]_\sigma \in \mathbb{F}_3$.
- Party σ gets $\mathbf{M}^{\bar{\sigma}}_{[\![\alpha_i]\!]_{\bar{\sigma}}}[[\![\alpha_i]\!]_\sigma] \in \{0,1\}^{3(\lambda+1)}$ while party $\bar{\sigma}$ gets nothing.

4: Define $[\![\mathsf{CW}_i]\!]_\sigma := \mathbf{M}^{\bar{\sigma}}_{[\![\alpha_i]\!]_{\bar{\sigma}}}[[\![\alpha_i]\!]_\sigma] \oplus z^\sigma$ and broadcast $[\![\mathsf{CW}_i]\!]_\sigma$.

5: Construct $\mathsf{CW}_i := [\![\mathsf{CW}_i]\!]_\sigma \oplus [\![\mathsf{CW}_i]\!]_{\bar{\sigma}} \in \{0,1\}^{3(\lambda+1)}$.

6: Output $(\mathsf{CW}_{i,0}, \mathsf{CW}_{i,1}, \mathsf{CW}_{i,2})$.

Other Optimizations of FOLEAGE

Using a single multi-evaluation step

- Alice | Bob evaluates $f^{A|B} = \left\langle a \otimes a, (e_0 \otimes e_1)^{A|B} \right\rangle$, $x[g] \cdot y[g] = f^A(g) + f^B(g)$ for $g \in G$
- Instead of FFT $\rightarrow$ IFFT $\rightarrow$ FFT, we can keep FFT($a \otimes a$) as pp and perform only one FFT

FFT Optimization

- Recall that order $|G| = 3^n$
- Full-evaluation is traversing on a **tenary** tree
- Use classic divide-and-conquer algorithm to achieve $O(n3^n)$ complexity

$$P(X_1, \ldots, X_n) = P_0(X_1, \ldots, X_{n-1}) + X_n P_1(X_1, \ldots, X_{n-1}) + X_n^2 P_2(X_1, \ldots, X_{n-1})$$

- $\text{work}(n) = 3 \cdot \text{work}(n-1) + 2 \cdot 3^n$
- $\text{work}(n) = 2 \cdot n \cdot 3^n$

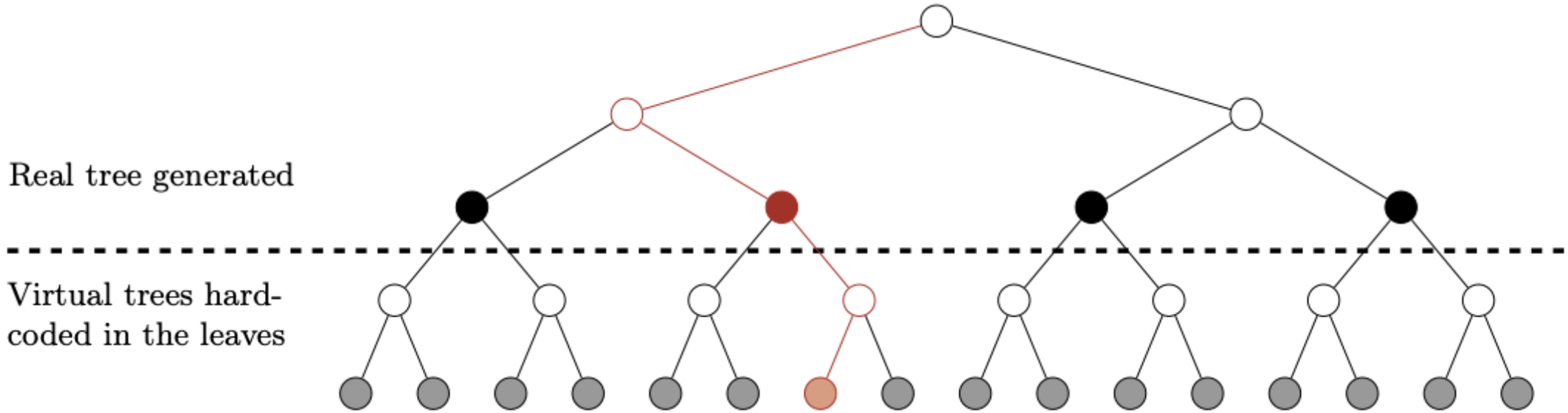
Additional FFT Optimization

- Recall that there are c^2 polynomials in $e_0 \otimes e_1$
- We can pack 32 monomial evaluation in a 64-bit machine word
- Polynomial evaluation is XOR of monomial evaluations $\rightarrow$ 32-times optimization

Optimization with Early Termination

General Idea

- For FSS with small output domain, we can pack the truth table of a sub-tree in an internal node.



Problem

- Since the index is tenary, we can only pack $3^{\lceil \log_3(64) \rceil}$ leaves

Protocol $\Pi_{\text{Output-CW}}$

PARAMETERS:

- There are two parties $\sigma, \bar{\sigma} \in \{0,1\}$ with input $([\![\alpha_i]\!]_\sigma)_{i\in[t]} \in (\mathbb{F}_3)^t, [\![\beta]\!]_\sigma \in \mathbb{F}_4, s^\sigma \in \{0,1\}^\lambda$.
- An instantiation of chosen $\binom{1}{3}$-OT.
- Pseudorandom function $G\colon \{0,1\}^\lambda \to (\mathbb{F}_4)^{3^t}$.

PROTOCOL:

For each party $\sigma \in \{0,1\}$, for $i \in [t]$:

1: Sample $z_i^\sigma \leftarrow_R (\mathbb{F}_4)^{3^i}$.

2: Define

$$\mathbf{C}_{i,0}^\sigma = ([\![\beta]\!]_\sigma, 0, 0) \oplus z_i^\sigma \in (\mathbb{F}_4)^{3^i},$$
$$\mathbf{C}_{i,1}^\sigma = (0, [\![\beta]\!]_\sigma, 0) \oplus z_i^\sigma \in (\mathbb{F}_4)^{3^i},$$
$$\mathbf{C}_{i,2}^\sigma = (0, 0, [\![\beta]\!]_\sigma) \oplus z_i^\sigma \in (\mathbb{F}_4)^{3^i},$$
$$\mathbf{M}_0^\sigma = (\mathbf{C}_{i,0}^\sigma, \mathbf{C}_{i,1}^\sigma, \mathbf{C}_{i,2}^\sigma),\ \mathbf{M}_1^\sigma = (\mathbf{C}_{i,1}^\sigma, \mathbf{C}_{i,2}^\sigma, \mathbf{C}_0^\sigma),\ \mathbf{M}_2^\sigma = (\mathbf{C}_{i,2}^\sigma, \mathbf{C}_{i,0}^\sigma, \mathbf{C}_{i,1}^\sigma)$$

3: Invoke $\binom{1}{3}$-OT with party $\bar{\sigma}$ as follows:

- Party $\bar{\sigma}$ plays the role of the sender with inputs $\mathbf{M}^{\bar{\sigma}}_{[\![\alpha_i]\!]_{\bar{\sigma}}}$.
- Party σ plays the role of the receiver and inputs $[\![\alpha_i]\!]_\sigma \in \mathbb{F}_3$.
- Party σ gets $\mathbf{M}^{\bar{\sigma}}_{[\![\alpha_i]\!]_{\bar{\sigma}}}[[\![\alpha_i]\!]_\sigma] \in (\mathbb{F}_4)^{3^i}$ while party $\bar{\sigma}$ gets nothing.

4: Define $[\![\beta]\!]_\sigma := \mathbf{M}_i^{\bar{\sigma}}[[\![\alpha_i]\!]_\sigma] \oplus z_i^\sigma \in (\mathbb{F}_4)^{3^i}$.

Output $[\![\mathsf{CW}]\!]_t := [\![\beta]\!]_\sigma \oplus G(s^\sigma)$.

Converting Half-Tree Techniques to Tenary Trees

- Currently the 1-out-of-3 OT seems hard to instantiate using the half-tree technique
- The main difficulty, in my opinion, is how to express CW_i as a linear function on index α_i and its authentication
- In Half-tree, $CW_i = H(s_{i-1}^0) \oplus H(s_{i-1}^1) \oplus (1 \oplus \alpha_i) \cdot \Delta$
- In FOLEAGE,

$$\begin{aligned} CW_i = (&G_0(s_{i-1}^0) \oplus G_0(s_{i-1}^1) \oplus \mathbb{I}(\alpha_i = 0) \cdot r \| \\ &G_1(s_{i-1}^0) \oplus G_1(s_{i-1}^1) \oplus \mathbb{I}(\alpha_i = 1) \cdot r \| \\ &G_2(s_{i-1}^0) \oplus G_2(s_{i-1}^1) \oplus \mathbb{I}(\alpha_i = 2) \cdot r) \end{aligned}$$

Minor Details

- Index authentication over $\mathbb{F}_3$
- Tenary Half Tree

Distributed KeyGen for Half-Tree

Protocol Π_{DPF}

This protocol invokes $\Pi_{\mathsf{BatchCheck}}$ (Figure 2) as a sub-protocol.

Initialize: For each $b \in \mathbb{F}_2$, P_b samples $\Delta_b \leftarrow \mathbb{F}_{2^\lambda}$ such that $\mathsf{lsb}(\Delta_b) = b$, and sends $(\mathsf{init}, b, \Delta_b)$ to $\mathcal{F}_{\mathsf{aBit}}$.

Protocol inputs: Two parties P_0 and P_1 hold n BDOZ-style authenticated sharings $\langle\!\langle \alpha^{(i)} \rangle\!\rangle = (\langle\!\langle \alpha^{(i)} \rangle\!\rangle_0, \langle\!\langle \alpha^{(i)} \rangle\!\rangle_1)$ for all $i \in [0, n)$ as well as a SPDZ-style authenticated sharing $[\![\beta]\!] = ([\![\beta]\!]_0, [\![\beta]\!]_1)$. Let $N = 2^n$ for some $n \in \mathbb{N}$. Let $\mathcal{H}_0 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a CCR hash function and $\mathcal{H}_1 : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ such that $\mathcal{H}_1(x) := \mathcal{H}_0(x) \,\|\, \mathcal{H}_0(x \oplus 1)$.

Generate SPDZ-style authenticated sharings of DPF outputs: Let $\langle\!\langle \alpha^{(i)} \rangle\!\rangle_b = (\alpha_b^{(i)}, \mathsf{K}_b[\alpha_{1-b}^{(i)}], \mathsf{M}_b[\alpha_b^{(i)}])$ and $[\![\beta]\!]_b = (\beta_b, \mathsf{M}_b[\beta])$ for each $b \in \{0,1\}$. The parties P_0 and P_1 do the following.

1. Both parties call $\mathcal{F}_{\mathsf{coin}}$ to sample a public randomness $W \in \mathbb{F}_{2^\lambda}$. Each party P_b sets $(s_b^{(0,0)} \,\|\, t_b^{(0,0)}) := \Delta_b \oplus W \in \{0,1\}^\lambda$.
2. For each $b \in \{0,1\}$, for each $i \in [0, n)$, P_b computes the following:

$$\mathsf{CW}_b^{(i)} := \left(\bigoplus\nolimits_{j \in [0,2^i)} \mathcal{H}_0(s_b^{(i,j)} \,\|\, t_b^{(i,j)})\right) \oplus \Delta_b \oplus \left(\alpha_b^{(i)} \cdot \Delta_b \oplus \mathsf{K}_b[\alpha_{1-b}^{(i)}] \oplus \mathsf{M}_b[\alpha_b^{(i)}]\right) \in \{0,1\}^\lambda,$$

and sends $\mathsf{CW}_b^{(i)}$ to P_{1-b}. For each $i \in [0, n)$, both parties compute $\mathsf{CW}^{(i)} := \mathsf{CW}_0^{(i)} \oplus \mathsf{CW}_1^{(i)}$, and each party P_b computes:

$$\left(s_b^{(i+1,2j)} \,\|\, t_b^{(i+1,2j)}\right) := \mathcal{H}_0\left(s_b^{(i,j)} \,\|\, t_b^{(i,j)}\right) \oplus t_b^{(i,j)} \cdot \mathsf{CW}^{(i)} \text{ for each } j \in [0, 2^i),$$

$$\left(s_b^{(i+1,2j+1)} \,\|\, t_b^{(i+1,2j+1)}\right) := \mathcal{H}_0\left(s_b^{(i,j)} \,\|\, t_b^{(i,j)}\right) \oplus \left(s_b^{(i,j)} \,\|\, t_b^{(i,j)}\right) \oplus t_b^{(i,j)} \cdot \mathsf{CW}^{(i)} \text{ for each } j \in [0, 2^i).$$

Distributed KeyGen for Half-Tree (Continued)

3. For each $b \in \{0,1\}$, P_b computes

$$\mathsf{CW}_b^{(n)} := \left(\bigoplus_{j\in[0,N)} \mathcal{H}_1(s_b^{(n,j)} \parallel t_b^{(n,j)})\right) \oplus (\beta_b \parallel \mathsf{M}_b[\beta]) \in \{0,1\}^{2\lambda},$$

and sends $\mathsf{CW}_b^{(n)}$ to P_{1-b}. Then, both parties compute $\mathsf{CW}^{(n)} := \mathsf{CW}_0^{(n)} \oplus \mathsf{CW}_1^{(n)}$. For each $b \in \{0,1\}$, P_b computes

$$[\![u^{(j)}]\!]_b := \left(u_b^{(j)} = t_b^{(n,j)}, \mathsf{M}_b[u^{(j)}] = (s_b^{(n,j)} \parallel t_b^{(n,j)})\right) \text{ for each } j \in [0,N),$$
$$[\![v^{(j)}]\!]_b = \left(v_b^{(j)} \parallel \mathsf{M}_b[v^{(j)}]\right) := \mathcal{H}_1\left(s_b^{(n,j)} \parallel t_b^{(n,j)}\right) \oplus t_b^{(n,j)} \cdot \mathsf{CW}^{(n)} \text{ for each } j \in [0,N).$$

4. As in the **Rand** process of protocol Π_{2PC} (Figure 4), both parties call functionality $\mathcal{F}_{\mathsf{aBit}}$ to generate $[\![r]\!]$ with a random $r \in \mathbb{F}_{2^\lambda}$. Then, both parties call functionality $\mathcal{F}_{\mathsf{coin}}$ to sample a random challenge $\chi \in \mathbb{F}_{2^\lambda}$, and locally compute

$$[\![a]\!] := \sum_{j\in[0,N)} \chi^j \cdot [\![u^{(j)}]\!] + \sum_{j\in[0,N)} \chi^{N+j} \cdot [\![v^{(j)}]\!] + [\![r]\!].$$

5. As in the **Open** process of protocol Π_{2PC}, both parties open $[\![a]\!]$ to obtain $\tilde{a} = a_0 + a_1 \in \mathbb{F}_{2^\lambda}$ by letting P_0 send a_0 to P_1 and P_1 send a_1 to P_0 in parallel. Then, both parties run sub-protocol $\Pi_{\mathsf{BatchCheck}}$ (Figure 2) on input $([\![a]\!], \tilde{a})$ to check $a = \tilde{a}$.
6. For each $j \in [0,N)$, both parties obtain $[\![u^{(j)}]\!] = ([\![u^{(j)}]\!]_0, [\![u^{(j)}]\!]_1)$ and $[\![v^{(j)}]\!] = ([\![v^{(j)}]\!]_0, [\![v^{(j)}]\!]_1)$.

Some Confusing Points

What's the cost of broadcast

- $P_2, \ldots, P_n$ sends shares to P_1, who sends back reconstructed value
- Total comm. is $2(n-1)$ bits, amortized comm.≈ 2 bits

Protocol $\Pi_{\mathsf{BT}}(\mathbb{F}_4 \to \mathbb{F}_2)$

PROTOCOL:

1: The parties invoke the functionality $\mathcal{F}_{\mathsf{cBT}}(\mathbb{F}_4)$ with init. Each party P_i receives a triple $([\![a]\!]_i^4, [\![b]\!]_i^4, [\![c]\!]_i^4) \in \mathbb{F}_4^3$.

2: Each party P_i broadcasts $[\![b]\!]_i^4(1)$. All parties reconstruct $b(1) = \sum_{i=1}^{N} [\![b]\!]_i^4(1)$.

OUTPUT: Each party P_i outputs $([\![a]\!]_i^4(0), [\![b]\!]_i^4(0), [\![c]\!]_i^4(0) + b(1) \cdot [\![a]\!]_i^4(1))$.

Lemma 21. *The protocol $\Pi_{\mathsf{BT}}(\mathbb{F}_4 \to \mathbb{F}_2)$ of Fig. 16 securely realizes the $\mathcal{F}_{\mathsf{cBT}}(\mathbb{F}_2)$ corruptible functionality in the $\mathcal{F}_{\mathsf{cBT}}(\mathbb{F}_4)$-hybrid model, using one bit of communication per party and a single call to $\mathcal{F}_{\mathsf{cBT}}(\mathbb{F}_4)$.*

What's the cost of GMW online

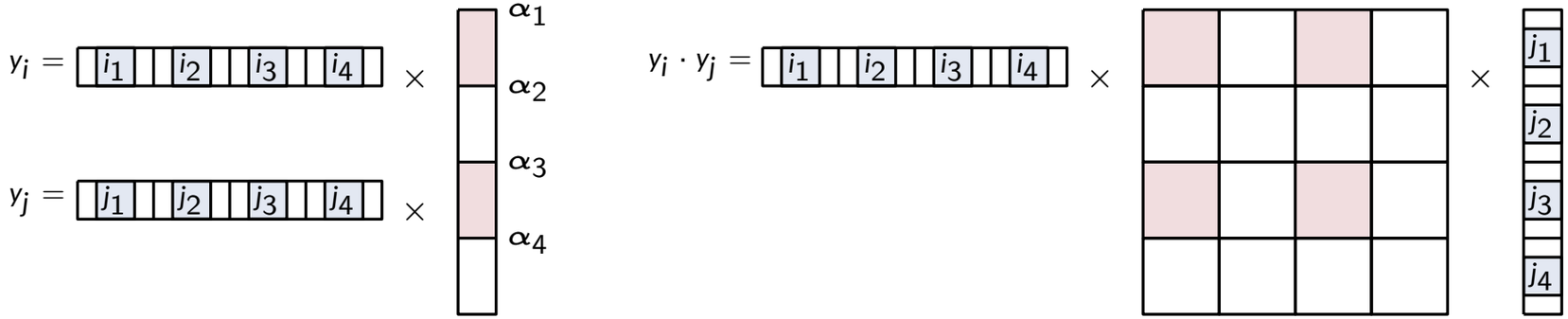
- With star-sharing, 1 broadcast suffices
- With additive sharing, we need 2 broadcasts

Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, ... polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = h_j^T \cdot e$
- Therefore, $y_i \cdot y_j = (\mathbf{h_i} \otimes \mathbf{h_j})^T \cdot (\mathbf{e} \otimes \mathbf{e})$

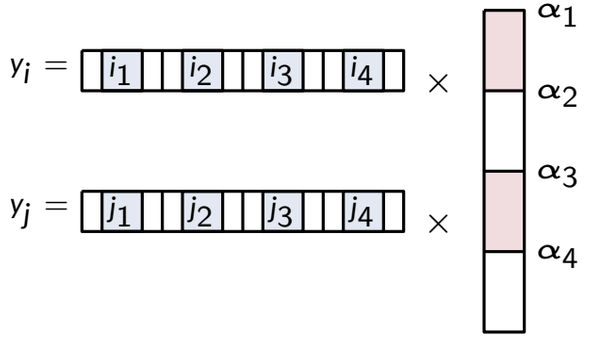
Garbling Pseudorandom Correlation Function

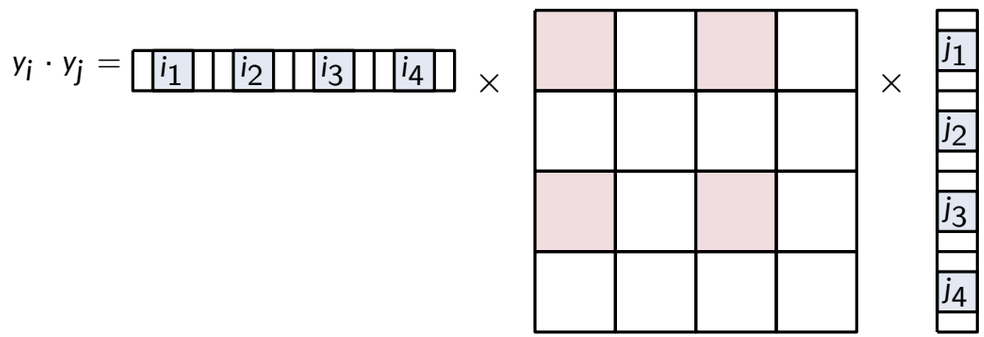
- Using FSS for decision tree, we can support degree-2, 3, … polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = h_j^T \cdot e$
  (c-sparse, t-sparse)
- Therefore, $y_i \cdot y_j = (\mathbf{h_i} \otimes \mathbf{h_j})^T \cdot (\mathbf{e} \otimes \mathbf{e})$
  (c^2-sparse, t^2-sparse)
- With FSS for decision trees, we can support constant degree polynomial evaluation over the **y** coordinates.



Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, … polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = h_j^T \cdot e$
  (h_i: c-sparse, e: t-sparse)
- Therefore, $y_i \cdot y_j = (\mathbf{h_i} \otimes \mathbf{h_j})^T \cdot (\mathbf{e} \otimes \mathbf{e})$
  ($\mathbf{h_i} \otimes \mathbf{h_j}$: c^2-sparse, $\mathbf{e} \otimes \mathbf{e}$: t^2-sparse)
- With FSS for decision trees, we can support constant degree polynomial evaluation over the **y** coordinates.





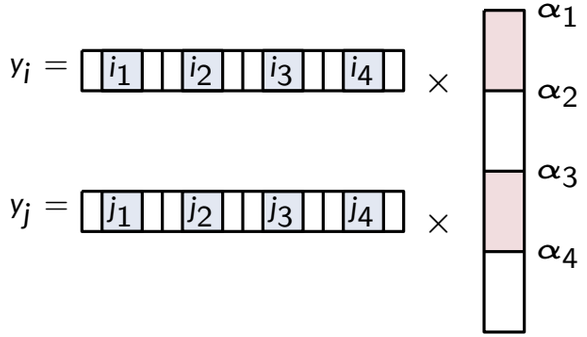
$$y_i \cdot y_j = \mathrm{sh}_{1,1} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,2} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases}$$

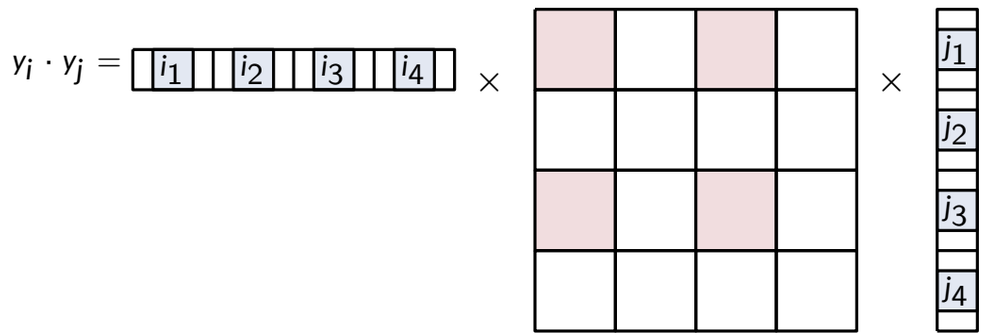
$$+ \mathrm{sh}_{1,3} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,4} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases}$$

$$+ \ldots$$

Garbling Pseudorandom Correlation Function

- Using FSS for decision tree, we can support degree-2, 3, … polynomials
- Recall that with EA-LPN, we have $y_i = h_i^T \cdot e$, $y_j = h_j^T \cdot e$
  (c-sparse, t-sparse)
- Therefore, $y_i \cdot y_j = (\mathbf{h_i} \otimes \mathbf{h_j})^T \cdot (\mathbf{e} \otimes \mathbf{e})$
  (c^2-sparse, t^2-sparse)
- With FSS for decision trees, we can support constant degree polynomial evaluation over the **y** coordinates.

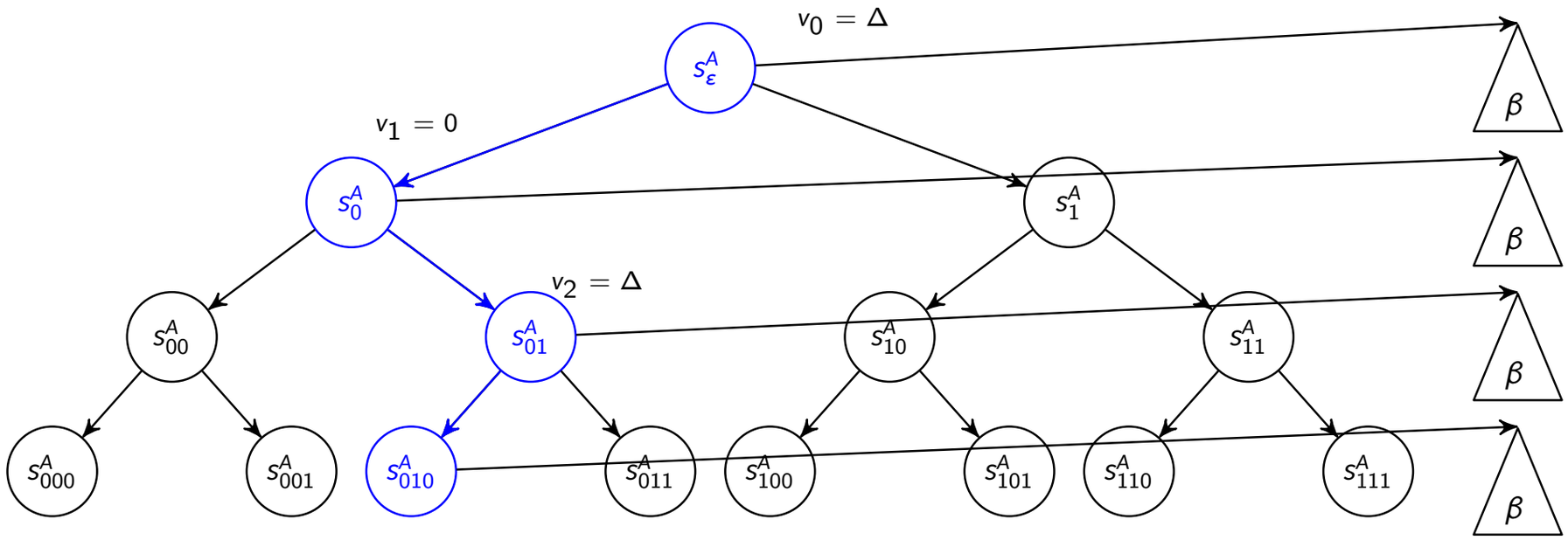




$$y_i \cdot y_j = \mathrm{sh}_{1,1} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_1 \geq \alpha_1 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,2} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_2 < \alpha_2 \\ 0 & \text{otherwise} \end{cases}$$
$$+ \mathrm{sh}_{1,3} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_3 \geq \alpha_3 \\ 0 & \text{otherwise} \end{cases} + \mathrm{sh}_{1,4} = \begin{cases} 1 & i_1 \geq \alpha_1 \wedge j_4 < \alpha_4 \\ 0 & \text{otherwise} \end{cases}$$
$$+ \ldots$$

FSS for (2,3,…)-dim rectangles

Half-Tree, $\alpha = 010, \beta = 110, LSB(\Delta) = 1$



$$f^{\geq,\geq}_{\alpha,\beta}(x,y) = \begin{cases} 1 & x \geq \alpha \wedge y \geq \beta \\ 0 & ow \end{cases}$$

$\Delta = s^A_\varepsilon \oplus s^B_\varepsilon$, assume wlog. $LSB(\Delta) = 1$

Note: ◯ = share of Δ, ◯ = share of 0^λ

- If starting node $s^A = s^B$ then output share is always 0
- Nodes on the same level can re-use the same pp.